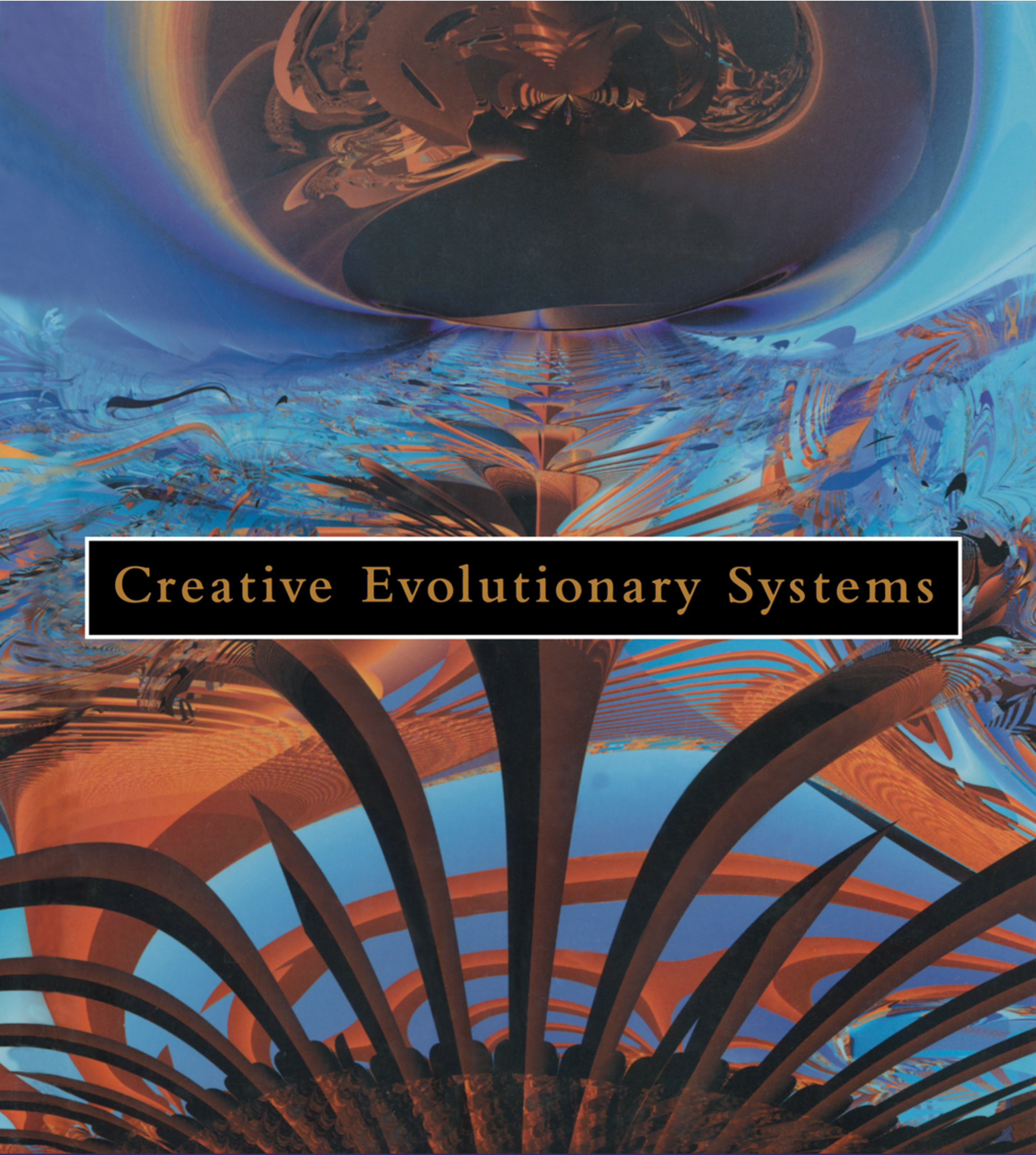
VISIT…

LANZAROTE
Caliente.COM



Creative Evolutionary Systems

EDITED BY

PETER J. BENTLEY & DAVID W. CORNE

Creative Evolutionary Systems

About the Editors

Peter J. Bentley is an Honorary Research Fellow at UCL, known for his prolific research covering all aspects of EC, including multiobjective optimization, constraint handling, artificial immune systems, computational embryology, and more, and applied to diverse applications including floor-planning, control, fraud detection, and music composition. He speaks regularly at international conferences and is a consultant, convenor, chair, and reviewer for workshops, conferences, journals, and books on evolutionary design and evolutionary computation. He has been guest editor of special issues on evolutionary design and creative evolutionary systems in journals and is the editor of the book *Evolutionary Design by Computers* (foreword by Richard Dawkins) and author of the popular science book *Digital Biology*.

David W. Corne lectures and consults in EC at the University of Reading. His early research on evolutionary timetabling (with Peter Ross) resulted in the first freely available and successful EC-based general timetabling program for educational and other institutions. Later EC work has been in protein folding, human genome research, medicine, scheduling, layout design, telecommunications, data mining, algorithm comparison issues, multiobjective optimization, and others. He is an associate editor of the *IEEE Transactions on Evolutionary Computation*, a founding coeditor of the *Journal of Scheduling*, on the Review Board of Applied Intelligence, and he appears on a host of international conference program committees. Other recent edited books include *New Ideas in Optimisation* (with Marco Dorigo and Fred Glover) and *Telecommunications Optimisation: Heuristic and Adaptive Techniques* (with Martin Oates and George Smith).

Creative Evolutionary Systems

Peter J. Bentley
Department of Computer Science
University College London

David W. Corne
School of Computer Science, Cybernetics and Electronic Engineering
University of Reading

Executive Editor Denise E. M. Penrose
Publishing Services Manager Scott Norton
Production Editor Howard Severson
Editorial Coordinator Emilia Thiuri
Production Assistant Mei Levenson
Cover Design Yvo Riezebos
Cover Image *Hyperspace Embryo.* © 1999 Steven Rooke. A true genetic fractal.
Text Design Windfall Software
Composition and Illustration Technologies 'N Typography
Copyeditor Ken DellaPenta
Proofreader Erin Milnes
Indexer Ty Koontz
Printer Courier Corporation

Designations used by companies to distinguish their products are often claimed as trademarks or registered trademarks. In all instances where Morgan Kaufmann Publishers is aware of a claim, the product names appear in initial capital or all capital letters. Readers, however, should contact the appropriate companies for more complete information regarding trademarks and registration.

Morgan Kaufmann Publishers
340 Pine Street, Sixth Floor, San Francisco, CA 94104–3205, USA
http://www.mkp.com

ACADEMIC PRESS
A Harcourt Science and Technology Company
525 B Street, Suite 1900, San Diego, CA 92101–4495, USA
http://www.academicpress.com

Academic Press
Harcourt Place, 32 Jamestown Road, London NW1 7BY, United Kingdom
http://www.academicpress.com

© 2002 by Academic Press
All rights reserved
Printed in the United States of America

06 05 04 03 02 5 4 3 2 1

No part of this publication may be reproduced, stored in a retrieval system, or transmitted in any form or by any means—electronic, mechanical, photocopying, recording, or otherwise—without the prior written permission of the publisher.

Library of Congress Cataloging-in-Publication Data is available for this book.
Library of Congress Control Number: 2001090636
ISBN: 1–55860–673–4

This book has been printed on acid-free paper.

Foreword

by Margaret A. Boden
University of Sussex

Computer creativity has been slipping onto our screens and edging into our consciousness for 40 years. Recently, however, a new approach has been added to the computer creator's armory: evolution. Programs using evolutionary algorithms can evolve unexpected structures, if necessary over many thousands of generations, of a form that the human mind could not have produced by itself. As this volume shows, these structures may lie in the visual, graphic, or musical arts, or in chemistry, engineering, or robotics. Broadly speaking, if you can program it, you can evolve it.

An evolutionary algorithm involves two aspects. First, a way of enabling the program to change its own rules, by making random variations in the current generation. These variations may be relatively superficial, such as altering the value of some numerical parameter in the structure-building code. Or they may be much more radical, such as inserting an entire new program into the heart of the current code. The genealogies, or family relationships, will be more readily visible in the former case. It's the latter case that sometimes comes up with big surprises.

The second aspect of an evolutionary algorithm is selection: there has to be some way of choosing, at each generation, those individuals who will be used to breed the next. And here, things can get even more difficult, and even more interesting.

More "difficult," because it's usually hard to identify our evaluative criteria even in intuitive terms, and very much harder to express them in computational form. Maybe we know how to choose between sibling molecules when we're trying to evolve a drug with a certain chemical effect. And maybe we can tell the program how to make that choice for itself. If the program can apply the fitness function, we can go off to the pub and leave it to do its own thing. But what about music, or visual art? Is there, or isn't there, any hope of doing the same in these domains?

More "interesting," because if we can't get the computer to do the selection automatically, we must do it ourselves. A whole host of novelties can be automatically generated, from which we choose interactively. We can instruct the program to breed from this pair, rather than that pair. Provided that the variations aren't too unconstrained, so that our preferences are already wiped out by the great-grandchildren stage, we can cooperate with the program in coming up with otherwise unimaginable structures, whose aesthetics are broadly guided—though never wholly controlled—by us.

This volume shows the current state of the art, and the science, of evolutionary creativity. It ranges over many different examples and shows what can—and, equally important, what can't—be done at the turn of the new millennium. What will have been achieved by the turn of the next one is anyone's guess. Meanwhile, it's intriguing, it's instructive, it's difficult, and it's fun.

To Anna Christina Catherine Corne
—DWC

This Page Intentionally Left Blank

Contents

Foreword v
By Margaret Boden

Contributors xxiii

Preface xxvii

An Introduction to Creative Evolutionary Systems 1
By Peter J. Bentley and David W. Corne

Introduction 1
AI and Creativity 2
Evolutionary Computation 4
Creative Evolutionary Systems 36
Is Evolution Creative? 55

PART I Evolutionary Creativity 77

1 Creativity in Evolution: Individuals, Interactions, and Environments 79
By Tim Taylor

1.1 **Introduction** 79
1.2 **Creativity and Opened-Ended Evolution** 79
1.3 **Design Issues** 82
    1.3.1 Von Neumann's Architecture for Self-Reproduction 82
    1.3.2 Tierra 84
    1.3.3 Implicit versus Explicit Encoding 87
    1.3.4 Ability to Perform Other Tasks 91
    1.3.5 Embeddedness in the Arena of Competition and Richness of Interactions 93
    1.3.6 Materiality 98
1.4 **A Full Specification For An Open-Ended Evolutionary Process** 100
    1.4.1 Waddington's Paradigm for an Evolutionary Process 101
1.5 **Conclusions** 104
Acknowledgments 105
References 105

2 Recognizability of the Idea: The Evolutionary Process of Argenia 109
By Celestino Soddu

2.1 **Introduction** 109
2.2 **Recognizability, Identity, and Complexity** 110
2.3 **Evolutionary Codes: Artificial DNA** 111
2.4 **Natural/Artificial Complexity** 112
2.5 **Giotto, a Medieval Idea in Evolution** 114
2.6 **Rome, Future Scenarios** 116
2.7 **Basilica, Generative Software to Design Complexity** 116
2.8 **Madrid and Milan, Generated Architecture** 119
2.9 **Argenìa, the Natural Industrial Object, and the Artificial Uniqueness of Species** 121
2.10 **Argenìc Art: Picasso** 123
2.11 **Conclusions** 125
References 127

3 Breeding Aesthetic Objects: Art and Artificial Evolution 129
By Mitchell Whitelaw

3.1 **Introduction** 129
3.2 **Breeding Aesthetic Objects** 130
3.2.1 A Case Study—Steven Rooke 131
3.3 **Breeding and Creation** 133
3.3.1 Creative Agency and the Breeding Process 134
3.3.2 The Evolved Aesthetic Object 136
3.4 **Limits** 137
3.5 **Driessens and Verstappen—an Alternative Approach** 139
3.6 **Conclusions** 144
References 144

4 The Beer Can Theory of Creativity 147
By Liane Gabora
4.1 **Introduction** 147
4.2 **Culture as an Evolutionary Process** 148
4.2.1 Variation and Convergence in Biology and Culture 148
4.2.2 Is More Than One Mind Necessary for Ideas to Evolve? 150
4.2.3 Meme and Variations: A Computer Model of Cultural Evolution 151
4.2.4 Breadth-First versus Depth-First Exploration 152
4.2.5 Dampening Arbitrary Associations and Forging Meaningful Ones 153
4.3 **Creativity as the Origin of Culture** 154
4.3.1 Theoretical Evidence 155
4.3.2 Archeological Evidence 155
4.3.3 Evidence from Animal Behavior 156
4.4 **What Caused the Onset of Creativity?** 156
4.5 **Conclusions** 158
Acknowledgments 159
References 159

PART II Evolutionary Music 163

5 GenJam: Evolution of a Jazz Improviser 165
By John A. Biles

5.1 **Introduction** 165
5.2 **Overview and Architecture** 166
5.3 **Representations** 168
5.4 **Genetic Operators and Training** 172
  5.4.1 Crossover 173
  5.4.2 Musically Meaningful Mutation 175
5.5 **Real-Time Interaction** 179
5.6 **Conclusions** 184
References 186

6 On the Origins and Evolution of Music in Virtual Worlds 189

By Eduardo Reck Miranda

6.1 **Introduction** 189
6.2 **Evolutionary Modeling** 190
  6.2.1 Transformation and Selection 191
  6.2.2 Coevolution 192
  6.2.3 Self-organization 192
  6.2.4 Level Formation 194
6.3 **Evolving Sound with Cellular Automata** 194
  6.3.1 The Basics of Cellular Automata 195
  6.3.2 The Cellular Automaton Used in Our System 196
  6.3.3 The Synthesis Engine 199
6.4 **Commentary on the Results** 201
6.5 **Conclusions** 202
Acknowledgments 202
References 203

7 Vox Populi: Evolutionary Computation for Music Evolution 205

By Artemis Moroni, Jônatas Manzolli, Fernando Von Zuben, and Ricardo Gudwin

7.1 **Introduction** 206
7.2 **Sound Attributes** 208
7.3 **Evolutionary Musical Cycle** 208
  7.3.1 The Voices Population 209
  7.3.2 The Rhythm of the Evolution 210

7.4 **Fitness Evaluation** 211
  7.4.1 The Consonance Criterion 212
  7.4.2 Melodic Fitness 214
  7.4.3 Harmonic Fitness 214
  7.4.4 Voice Range Criterion 215
  7.4.5 Musical Fitness 215
7.5 **Interface and Parameter Control** 216
7.6 **Experiments** 218
7.7 **Conclusions** 219
Acknowledgments 220
References 220

8 The Sound Gallery—An Interactive A-Life Artwork 223

By Sam Woolf and Adrian Thompson

8.1 **Introduction** 223
8.2 **Evolvable Hardware** 224
  8.2.1 Reconfigurable Chips 227
8.3 **Gallery Setup** 228
  8.3.1 Setting 228
  8.3.2 Sensing Systems 230
8.4 **Contextualization: Artificial Life and Art** 231
  8.4.1 Evolutionary Algorithms and Visual Arts 231
  8.4.2 Evolutionary Algorithms and Music 232
  8.4.3 Interactive Genetic Art 234
  8.4.4 Interactive, Adaptive, and Autonomous (Nongenetic) Artworks 235
8.5 **The Sound Gallery Algorithms** 237
  8.5.1 Two-Phase Hill-Climbing/ Island Model GA 238
  8.5.2 Hill-Climbing Phase 238
  8.5.3 Island Model Genetic Algorithm Phase 239
  8.5.4 The Need for Aging 239
  8.5.5 Encoding Scheme 240
  8.5.6 The Fitness Function 241
  8.5.7 galSim 242
8.6 **The Experiment** 242
  8.6.1 Results 244
8.7 **Conclusions** 247
Acknowledgments 248
References 248

PART III Creative Evolutionary Design 251

9 Creative Design and the Generative Evolutionary Paradigm 253

By John Frazer

9.1 **Introduction** 253

9.2 **The Adaptive Model from Nature** 255

9.3 **The Generative Evolutionary Paradigm** 255

9.4 **Problems with the Paradigm** 257

9.5 **Concept Seeding Approach** 259

9.6 **The Reptile Demonstration** 260

9.7 **Universal State Space Modeler** 264

9.8 **Logic Fields** 266

9.9 **Returning to the Analogy with Nature** 269

9.10 **Conclusions** 271

References 273

10 Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results 275

By John R. Koza, Forrest H. Bennett III, David Andre, and Martin A. Keane

10.1 **Introduction** 275

10.2 **Inventiveness and Creativity** 276

10.3 **Genetic Programming** 280

10.4 **Applying Genetic Programming to Circuit Synthesis** 283

- 10.4.1 Campbell 1917 Ladder Filter Patent 285
- 10.4.2 Zobel 1925 "*M*-Derived Half Section" Patent 286
- 10.4.3 Cauer 1934–1936 Elliptic Filter Patents 287
- 10.4.4 Amplifier, Computational, Temperature-Sensing, Voltage Reference, and Other Circuits 288

10.5 **Topology, Sizing, Placement, and Routing of Circuits** 289

10.6 **Automatic Synthesis of Controllers by Means of Genetic Programming** 290
  10.6.1 Robust Controller for a Two-Lag Plant 292
10.7 **The Illogical Nature of Creativity and Evolution** 294
10.8 **Conclusions** 296
References 296

11 Toward a Symbiotic Coevolutionary Approach to Architecture 299

By Helen Jackson

11.1 **Introduction** 299
11.2 **Lindenmayer Systems** 299
  11.2.1 Example L-Systems 300
  11.2.2 The Isospatial Grid 302
  11.2.3 Spatial Embryology 302
11.3 **Artificial Selection** 302
  11.3.1 The Eyeball Test 304
11.4 **Single-Goal Evolution** 305
  11.4.1 "Generic Function" as Fitness Function 306
  11.4.2 Evolution toward Low i-Values 307
  11.4.3 Structural Stability 307
  11.4.4 Architecture as a Multigoal Task 307
  11.4.5 Dual-Goal Evolution 309
11.5 **Representation, Systems, and Symbiosis** 309
  11.5.1 Coevolution 310
  11.5.2 Naïve Architectural Form Representation 310
  11.5.3 Spatial Embryology 311
11.6 **Conclusions** 311
Acknowledgments 312
References 312

12 Using Evolutionary Algorithms to Aid Designers of Architectural Structures 315

By Peter von Buelow

12.1 **Introduction** 315
12.2 **Analysis Tools vs. Design Tools** 316
12.3 **Advantages of Evolutionary Systems in Design** 317

12.3.1 Use of Populations 317
12.3.2 Recombination and Mutation 318
12.3.3 Wide Search of Design Space 318
12.3.4 No Knowledge of the Objective Function 319
12.3.5 Imitation of Human Design Process 319
12.3.6 Can Learn from Designer 319

12.4 **Characteristics of an IGDT** 320
12.4.1 Definition of the IGDT Concept 320
12.4.2 Relation of IGDT to Design Process 322

12.5 **Mechanics of an IGDT** 323

12.6 **IGDT Operation** 328
12.6.1 Problem Definition 328
12.6.2 Initial IGDT Generation 329
12.6.3 Initial Generation with Designer Selection/Interaction 330
12.6.4 Second-Generation IGDT Response 331
12.6.5 Second-Generation Designer Interaction 332
12.6.6 Third Generation 332

12.7 **Conclusions** 335

Acknowledgments 335

References 335

PART IV Evolutionary Art 337

13 Eons of Genetically Evolved Algorithmic Images 339
By Steven Rooke

13.1 **Introduction** 339

13.2 **Using GP for Art** 339
13.2.1 Genetic Variation 340
13.2.2 Genetic Library 344
13.2.3 Functions and Node Internals 347
13.2.4 A Typical Run 348

13.3 **Horizon Lines and Fantasy Landscapes** 351

13.4 **Genetic Fractals** 351
13.4.1 Second-Order Subtleties of Orbit Trajectories during Iteration in the Complex Plane 358

13.5 **The Genetic Cross Dissolve** 358

13.6 **What Is It?** 360
    13.6.1 Constraints of Color and Form 361
    13.6.2 A Joyride for the Visual Cortex? 363
    13.6.3 Approaching the Organic 364
13.7 **Conclusions** 364
References 365

14 Art, Robots, and Evolution as a Tool for Creativity 367
By Luigi Pagliarini and Henrik Hautop Lund
14.1 **Introduction** 367
14.2 **The Social Context of Electronics** 368
    14.2.1 Where Electronics Acts 368
    14.2.2 How Technology Influences Art (the World) 369
    14.2.3 How Technology Gets Feedback (from Art and the World) 370
14.3 **What Artist?** 370
    14.3.1 Two Different Concepts or Aspects of the Artist 370
    14.3.2 Art and Human Language: The "Immaterial" Artist 371
    14.3.3 Art and Human Technique: The "Material" Artist 371
14.4 **Electronic Art** 372
    14.4.1 A New Electronic Space 373
    14.4.2 The "Material" Electronic Artist 373
    14.4.3 The "Immaterial" Artist and the Uses of Electronics 374
    14.4.4 Example—The Artificial Painter 375
14.5 **Alive Art** 379
    14.5.1 Other Artistic Movements Based on Electronics 379
    14.5.2 Alive Art 380
    14.5.3 The Aliver 381
    14.5.4 The "Alive Art Effect" 382
    14.5.5 Example—LEGO Robot Artists 383
14.6 **Conclusions** 384
References 384

15 Stepping Stones in the Mist 387
By Paul Brown
15.1 **Introduction** 387
15.2 **On My Approach as an Artist—a Disclaimer** 387
15.3 **Major Influences** 389
15.4 **Historical Work—1960s and 1970s** 392

15.5 **Early Computer Work** 396
15.6 **Recent Work** 402
15.7 **Current and Future Directions** 405
15.8 **Conclusions** 405
Acknowledgments 406
References 407

16 Evolutionary Generation of Faces 409
By Peter J. B. Hancock and Charlie D. Frowd
16.1 **Introduction** 409
  16.1.1 Eigenfaces 409
  16.1.2 Evolutionary Face Generator System 412
16.2 **Testing** 415
  16.2.1 Apparatus 415
  16.2.2 Generation of Face Images 415
  16.2.3 Evolutionary Algorithm 416
  16.2.4 Participants 417
16.3 **Results** 417
16.4 **Discussion** 421
16.5 **Conclusions** 422
Acknowledgments 423
References 423

17 The Escher Evolver: Evolution to the People 425
By A. E. Eiben, R. Nabuurs, and I. Booij
17.1 **Introduction** 425
17.2 **The Mathematical System behind Escher's Tiling** 427
17.3 **Evolutionary Algorithm Design** 428
  17.3.1 Representation 429
  17.3.2 Ground Shape and Transformation System 429
  17.3.3 Genetic Operators: Mutation and Crossover 432
  17.3.4 Selection Mechanism 433
17.4 **Implementation and the Working of the System** 434
  17.4.1 Stand-Alone Version 434
  17.4.2 First Networked Version 434
  17.4.3 Second Networked Version 435
17.5 **Conclusions** 437

Acknowledgments 439
References 439

PART V Evolutionary Innovation 441

18 The Genetic Algorithm as a Discovery Engine: Strange Circuits and New Principles 443
By Julian F. Miller, Tatiana Kalganova, Dominic Job, and Natalia Lipnitskaya

18.1 **Introduction** 443
18.2 **The Space of All Representations** 445
18.3 **Evolutionary Algorithms that Assemble Electronic Circuits from a Collection of Available Components** 447
  18.3.1 Binary Circuit Symbols 448
  18.3.2 Multiple-Valued Circuits 449
18.4 **Results** 450
  18.4.1 One-Bit Adder 450
  18.4.2 Two-Bit Adder 452
  18.4.3 Two-Bit Multiplier 454
  18.4.4 Three-Bit Multiplier 457
  18.4.5 Multiple-Valued One-Digit Adder with Carry 459
18.5 **Fingerprinting and Principle Extraction** 462
18.6 **Conclusions** 464
References 465

19 Discovering Novel Fighter Combat Maneuvers: Simulating Test Pilot Creativity 467
By R. E. Smith, B. A. Dike, B. Ravichandran, A. El-Fallah, and R. K. Mehra

19.1 **Introduction** 467
19.2 **Fighter Aircraft Maneuvering** 469
19.3 **Genetics-Based Machine Learning** 471
  19.3.1 Learning Classifier Systems 472
  19.3.2 The LCS Used Here 473

19.4 **"One-Sided Learning" Results** 479
19.5 **"Two-Sided Learning" Results** 480
19.6 **Differences in Goals and Techniques** 482
  19.6.1 Implications of This Goal 483
19.7 **Conclusions** 483
Acknowledgments 485
References 485

20 Innovative Antenna Design Using Genetic Algorithms 487

By Derek S. Linden

20.1 **Introduction** 487
20.2 **Antenna Basics** 489
20.3 **Conventional Designs and Unconventional Applications: The Yagi-Uda Antenna** 493
20.4 **Unconventional Designs and Conventional Applications: Crooked-Wire And Treelike Genetic Antennas** 500
  20.4.1 The Crooked-Wire Genetic Antenna 501
  20.4.2 Treelike Genetic Antennas 506
20.5 **Conclusions** 509
References 510

21 Evolutionary Techniques in Physical Robotics 511

By Jordan B. Pollack, Hod Lipson, Sevan Ficici, Pablo Funes, Greg Hornby, and Richard A. Watson

21.1 **Introduction** 511
21.2 **Coevolution** 512
21.3 **Research Thrusts** 513
21.4 **Evolution in Simulation** 514
21.5 **Buildable Simulation** 515
21.6 **Evolution and Construction of Electromechanical Systems** 517
21.7 **Embodied Evolution** 518
21.8 **Conclusions** 519
Acknowledgments 520
References 520

22 Patenting Evolved Bactericidal Peptides 525
By Shail Patel, Ian Stott, Manmohan Bhakoo, and Peter Elliott

22.1 **Introduction** 525
22.2 **Design Cycle** 526
22.3 **Hypothesis: Mechanism of Action** 528
22.4 **Experimental Measures and Modeling Techniques** 529
    22.4.1 Molecular Modeling 531
    22.4.2 Neural Networks 534
22.5 **Evolution** 536
22.6 **Patent Application** 537
    22.6.1 Comparing Patent Spaces 539
22.7 **Conclusions** 542
References 543

Index 547

The color plate section lies between pages 192 and 193.

This Page Intentionally Left Blank

Contributors

David Andre
Division of Computer Science
University of California
Berkeley, California 94720
dandre@cs.berkeley.edu

Forrest H. Bennett III
Genetic Programming Inc.
Box 1669
Los Altos, California 94023
forrest@evolute.com
(Now affiliated with Fuji Xerox Palo Alto Laboratories, Palo Alto, California)

Peter J. Bentley
Department of Computer Science
University College London
Gower Street
London WC1E 6BT
UK
P.Bentley@cs.ucl.ac.uk

Manmohan Bhakoo
Unilever Research
Port Sunlight Laboratory
Bebington, Wirral
UK
mammohan.bhakoo@unilever.com

John A. Biles
Department of Information Technology
Rochester Institute of Technology
Rochester, New York
jab@it.rit.edu

I. Booij
Leiden Institute of Advanced Computer Science
Leiden, The Netherlands
booij@liacs.nl

Paul Brown
P.O. Box 3603
South Brisbane QLD 4101
Australia
paul@paul-brown.com
http://www.paul-brown.com

David W. Corne
Department of Computer Science
University of Reading
Whiteknights
Reading RG6 6AY
UK
D.W.Corne@reading.ac.uk

B. A. Dike
The Boeing Company
St. Louis, Missouri
bruce.a.dike@boeing.com

A. E. Eiben
Computational Intelligence Group
Division of Mathematics and Computer Science
Free University Amsterdam
The Netherlands
gusz@cs.vu.nl

Peter Elliott
Unilever Research
Port Sunlight Laboratory
Bebington, Wirral
UK
peter.elliott@unilever.com

A. El-Fallah
Scientific Systems
Woburn, Massachusetts 01801
adel@ssci.com

Sevan Ficici
Computer Science Department
Brandeis University
Waltham, Massachusetts 02454

John Frazer
Swire Chair and Head of School
DTRC, School of Design
The Hong Kong Polytechnic University
Hung Hom, Kowloon
Hong Kong
sdfrazer@polyu.edu.hk

Charlie D. Frowd
Department of Psychology
University of Stirling
Stirling FK9 4LA
UK
cdf1@stir.ac.uk

Pablo Funes
Computer Science Department
Brandeis University
Waltham, Massachusetts 02454
pablo@cs.brandeis.edu

Liane Gabora
Center Leo Apostel, Vrije Universiteit Brussel
Krijgskundestraat 33
1160 Brussels, Belgium
http://www.vub.ac.be/CLEA/liane/
lgabora@vub.ac.be

Ricardo Gudwin
University of Campinas
Electrical Engineering Faculty
State Univeristy of Compinas
São Paulo
Brazil
gudwin@dca.fee.unicamp.br

Peter J. B. Hancock
Department of Psychology
University of Stirling
Stirling FK9 4LA
UK
pjbh1@stir.ac.uk

Greg Hornby
Computer Science Department
Brandeis University
Waltham, Massachusetts 02254

Helen Jackson
Coevolution
helen@coevolution.com

Dominic Job
School of Computing
Napier University
219 Colinton Road
Edinburgh EH14 1DJ
UK
d.job@dcs.napier.ac.uk

Tatiana Kalganova
Electronic and Computing Engineering Department
Brunel University
Uxbridge
Middlesex UB8 3PH
UK
tatiana.kalganova@brunel.ac.uk

Martin A. Keane
Econometrics Inc.
111 E. Wacker Dr.
Chicago, Illinois 60601
makeane@ix.netcom.com

John R. Koza
Stanford Medical Informatics
Department of Medicine
School of Medicine
Department of Electrical Engineering
School of Engineering
Stanford University
Stanford, California 94305
koza@stanford.edu
http://www.smi.stanford.edu/people/koza

Derek S. Linden
Linden Innovation Research LLC
P.O. Box 1601
Herndon, VA 20171–1601
dlinden@lir-llc.com

Natalia Lipnitskaya
Department of Computing
State University of Informatics and Radioelectronics
6 P.Brovky
Minsk
Belarus 220 600
nat.lip@usa.com

Hod Lipson
Computer Science Department
Brandeis University
Waltham, Massachusetts 02454

Henrik Hautop Lund
LEGO Lab
University of Aarhus, Aabogade 34
8200 Aarhus N.
Denmark
hhl@daimi.au.dk

Jônatas Manzolli
University of Campinas
Interdisciplinary Nucleus of Sound Communication
State University of Compinas
São Paulo
Brazil
jonatas@nics.unicamp.br

R. K. Mehra
Scientific Systems
Woburn, Massachusetts 01801
rkm@ssci.com

Julian F. Miller
School of Computer Science
University of Birmingham
Birmingham B15 2TT
UK
j.miller@cs.bham.ac.uk
http://www.sc.bham.ac.uk/~jfm/

Eduardo Reck Miranda
Sony Computer Science Laboratory—Paris
6 rue Amyot
75005 Paris, France
miranda@csl.sony.fr

Artemis Moroni
Technological Center for Informatics
The Automation Institute
State University of Compinas
São Paulo
Brazil
artermix@ia.cti.br

R. Nabuurs
Leiden Institute of Advanced Computer Science
Leiden, The Netherlands
rnabuurs@liacs.nl

Luigi Pagliarini
LEGO Lab
University of Aarhus, Aabogade 34
8200 Aarhus N.
Denmark
luigi@daimi.au.dk

Shail Patel
Unilever Research
Port Sunlight Laboratory
Bebington, Wirral
UK
shail.patel@unilever.com

Jordan B. Pollack
Computer Science Department
Brandeis University
Waltham, Massachusetts 02454
pollack@cs.brandeis.edu

B. Ravichandran
Scientific Systems
Woburn, Massachusetts 01801
ravi@ssci.com

Steven Rooke
srooke@azstarnet.com
http://www.azstarnet.com/~Srooke

R. E. Smith
The Intelligent Computing Systems Centre
The University of The West of England
Bristol, UK
robert.smith@uwe.ac.uk

Celestino Soddu
Director of Generative Design Lab
Department of Scienze de Territorio
Milan Polytechnic
Italy
http://soddu2.dst.polimi.it

Ian P. Stott
Unilever Research
Port Sunlight Laboratory
Bebington, Wirral
UK
ian.stott@unilever.com

Tim Taylor
Institute of Perception, Action and Behaviour
Division of Informatics
University of Edinburgh
Edinburgh
Scotland
tim.taylor@ed.ac.uk

Adrian Thompson
School of Cognitive and Computing Sciences
University of Sussex at Brighton
Falmer, Brighton BN1 9QH
UK
adrianth@cogs.susx.ac.uk

Peter von Buelow
University of Stuttgart
Institute for Lightweight Structures (IL)
Stuttgart, Germany
pvbuelow@il.bauingenieure.uni-stuttgart.de
pvbuelow@utkux.utk.edu

Fernando Von Zuben
University of Campinas
Electrical Engineering Faculty
State University of Compinas
São Paulo
Brazil
vonzuben@dca.fee.unicamp.br

Richard A. Watson
Computer Science Department
Brandeis University
Waltham, Massachusetts 02454
watson@cs.brandeis.edu

Mitchell Whitelaw
Faculty of Humanities and Social Sciences
University of Technology
Sydney
mitchell@symbiotic.org

Sam Woolf
75 Cholmeley Crescent
Highgate
London N6 5EX
www.alifeart.co.uk
woolf@altavista.net

Preface

PETER'S PREAMBLES

I am trying to be creative as I write this. The first part of any book should be attention grabbing, interesting, maybe a little different. So this is my attempt at creativity. But I have to admit that others may be better at it than me. Artists, musicians, designers, all these people are paid to be creative, to come up with novel and clever ideas, or to surprise, interest, and enthrall us. This mystical thing, this "creativity," is a very human activity. It is almost beyond analysis, and indeed some are superstitious enough to make sure they never think about how they achieve their accomplishments. And the accomplishments of creative people are numerous and impressive. From the variety of the arts, to the intricate and elegant design solutions tucked out of sight inside most of our products, we are a species capable of astonishing creativity.

But what has this to do with computers? Surely a silicon chip is the antithesis of creativity, the harbinger of dullness, the personification of the mundane. A computer will only dull the sharp edge of creativity, won't it? And the idea that computers might even be capable of creativity is an absurdity, isn't it?

Well, actually no.

Computers and creativity are truly coming together for the first time. As this book will show, evolutionary computation is expanding the capabilities and the role of computers. In addition to the "traditional AI" approaches, we now have evolutionary programs that help artists and musicians to be ever more creative. Our evolutionary techniques are being used to automatically find solutions to problems that traditionally required creative people. Using evolution, our computers are beginning to find inventive, novel, surprising, and exciting solutions. Creative evolutionary systems both help us to be creative and give the appearance of being creative themselves.

And finally—despite the fact that I am one of the developers of these techniques, I am still constantly surprised and excited by the output of our evolutionary software. To me it still feels slightly unbelievable that my computer is better at designing than me, better at composing music than me, and better at creating pieces of art than me, even though I developed the creative evolutionary systems that enable it to do these things. In many ways this still feels like science fiction, but it is, of course, science fact. I hope the chapters in this book will stimulate and inspire you, my creative reader, as they continue to inspire me.

DAVE'S DISCUSSIONS

I am being creative as I write this, according to my personal definition of creativity. Some readers may think differently, since this section started out as an exact copy of "Peter's Preambles." The plan was to write a personal preface paragraph, of similar length to Peter's, and with the same theme as Peter's, but conveying my own introductory views. In doing so I have already digressed much from the plan, but this is all perhaps illustrative of one of the key concepts occurring throughout the contributed chapters: novelty can arise from mutation of existing designs. As Peter notes, the first part of any book should be attention grabbing, interesting, and maybe a little different. As readers will note, most of that last sentence is revealingly unchanged from part of Peter's preambles. So this is my attempt at creativity. Let's all thank our lucky stars that many others are better at it than me.

But what about creative software? Surely computation concerns dull, mechanical, and repetitive activity. Computers can copy extremely well, but surely any variation they may introduce will more likely resemble a flat note than a novel theme, or a smudge rather than a novel perspective. Even if we allow for their programming to ensure that their automated mutations seems to "make sense," then surely their products still aren't creative—they're just doing what they're told! Well, yes and no. Yes, because computers simply obey the laws of physics, nothing more. Their output is the result of a determined dance of electrons subject to a variety of forces. No, because, in my opinion, exactly the same can be said of any of us. So, if I (or, more likely, Peter) can be called creative, then software is similarly capable of being so labeled. One of the more popular and successful ways in which machines are being coerced into creative result is by using evolutionary computation.

Although other techniques drawn from computer science and artificial intelligence have had considerable success with some subset of the arts, it seems justifiable to claim that only evolutionary computation is now making headway in *each* of music, visual art, and product design. Creative evolutionary systems are beginning to provide increasingly impressive support for the view that creativity can be scientifically explained.

WHAT'S IN THE BOOK

Although its sister title *Evolutionary Design by Computers* provided a few examples, this is the first book to focus on creative evolutionary systems. It brings together the best, the first, the most original, and of course the most creative people to work in the area. We have a wide and varied collection of topics covered in this volume and have grouped them together into five major parts entitled Evolutionary Creativity, Music, Design, Art, and Innovation. Many of the authors describe work that was first presented in the AISB Symposium on Creative Evolutionary Systems, organized by us during Easter 1999 for the Edinburgh Science Festival. But we have also invited as many of the other experts in this field as possible to showcase their work in this volume. Again, we must add a disclaimer: we have included as many chapters as we can, but for the sake of portability we must keep the number of pages finite. This means that although we have done our best to ensure a definitive collection of work, there will no doubt be omissions, for which we apologize.

Like its sister title, this book also contains a CD-ROM to accompany the text. Because of the "creative nature" of this work, you will find a number of more unusual features on this CD. As well as software toolkits from the leading experts in the field to help you begin in this area, demonstrations, tutorials, and examples of evolved art, the CD is playable on an audio CD player. For the first time, instead of just describing the output of evolutionary music systems, we can let you listen and judge for yourself.

The book has five major parts, intended to cover the major aspects of creative evolutionary systems.

The first part, "Evolutionary Creativity," describes and explores the relationships between creativity and evolution. The themes discussed in these chapters include what creativity is (or isn't), the relationship between evolution and creativity, whether or not evolution can enhance artists' creativity, and related issues. Each chapter explores these themes from a different viewpoint, ranging from cultural to technical.

The second part, "Evolutionary Music," provides examples of how evolutionary systems are used, often in collaboration with humans, to produce novel sounds, accompaniments, or short compositions. One of these chapters concerns a system that accompanies the author's live jazz concerts in real time, and which is a star of published music media. Another of the chapters in this section describes an interactive evolutionary system that can be used as a new form of musical instrument. The other chapters explore evolutionary concepts in the creation of novel forms of music in novel ways.

The third part, "Creative Evolutionary Design," provides a link to the sister book *Evolutionary Design by Computer.* This section explores the abilities of evolution to aid designers by generating novel and "creative" design solutions. The first chapter, written by a pioneering architect, argues strongly the case for evolutionary systems in design. The next chapter explores the subject of using GP to provide human-competitive results via new analog circuit designs. The third chapter investigates the use of Lindenmeyer systems with GP for evolving useful and novel architectural forms. The last chapter of this part shows how evolution can be used within an Intelligent Genetic Design Tool, enabling designers to explore new and creative solutions to engineering design problems.

The artistic side of creative evolutionary systems is shown by the fourth part, "Evolutionary Art." In this part, Steven Rooke—one of the best known of the current "evolutionary artists"—describes and illustrates how he has used GP to evolve some astonishing pieces of art. The next chapter, by the authors of the well-known Artificial Painter, focus on what evolutionary art means for art and artist. The third chapter, by artist and writer Paul Brown, describes his own motivation and inspiration as he uses methods such as cellular automata to create his work. Next, a very practical application for the methods used in evolutionary art is examined: the collaborative evolution of photorealistic faces. The final chapter in this part describes collaborative evolutionary art systems for generating works in the styles of Mondrian and Escher; the works of the "Escher Evolver" system described here are on display as part of an Escher exhibition at the City Museum, The Hague, at the time of writing.

The fifth and final part, "Evolutionary Innovation," examines a variety of further applications in which evolutionary systems have been employed to yield novel and surprising results. Chapters here describe the evolution of new kinds of electrical circuit, novel aircraft flight maneuvers, strange new antennae for reliable electromagnetic reception, linked control and morphology for robots and other physical devices, and new drugs with useful biological activity.

A FINAL NOTE

Continuing the tradition from its sister title, *Evolutionary Design by Computers*, this book also contains examples of the creativity of children. If you glance at the part opener pages, you will see five original pieces of art drawn by 10 and 11 year olds (selected during a competition held in North County Primary School, Colchester). In addition to providing a quirky contrast to the technical prose in this volume, the artwork demonstrates in a very graphic way just how creative human beings are. They may be young, but these talented children can still teach our evolutionary systems a thing or two about being artistic, inventive, and creative. Nevertheless, from the progress shown by the chapters in this volume, by the time these children begin their first jobs, creative evolutionary systems will have caught up with them. Enjoy the book!

ACKNOWLEDGMENTS

We would also both like to thank:

- Denise Penrose, her varied and ever-changing array of assistants, and all of the staff at MKP for their work in producing this book.
- The contributors to the book, who toiled tirelessly to produce another unbeatable collection of chapters.
- All the contributors to the CD-ROM, which will be entertaining and useful because of their efforts.
- Margeret Boden for writing the foreword.
- The five schoolchildren—Al Juby, Chelsey Baker, Rebecca Bell, Louise Sizer-James, and Clare Carson—from the North County Primary School in Colchester for providing the artwork on our part opener pages.
- Our friends, family, and colleagues for providing welcome distractions, support, and helpful suggestions.
- The careful and painstaking efforts of the proof-weeder.
- And finally (as usual) Peter would like to thank the cruel and indifferent, yet astonishingly creative process of natural evolution for providing the inspiration for his work. Long may it continue to do so.

This Page Intentionally Left Blank

An Introduction to Creative Evolutionary Systems

Peter J. Bentley University of College London
David W. Corne University of Reading

INTRODUCTION

In many ways, using the word "creativity" in a field of computer science opens up a Pandora's box of controversy. Convincing people that our creativity can be enhanced using computer software is sometimes difficult. Arguing with people that the techniques themselves might be capable of creativity is often futile. But there are an increasing number of researchers prepared to try.

These are the researchers who pioneer creative evolutionary systems. Their programs enable artists to evolve stunning pieces of art, or allow musicians to create new sounds and new compositions. By guiding evolution as it happens, the users are able to explore new ideas as they emerge through the mechanisms of evolution. Many find this a highly rewarding activity—exploring the space of possible images or sculptures or compositions, and being constantly surprised as possibilities are revealed to them. Other creative evolutionary systems take this approach one step further. Guidance is provided by automatic software routines that judge evolving solutions without the need of human input. Designs are evolved from random blobs to fully functional forms. Novel circuits, boat hulls, architectural forms, aircraft maneuvers, even chemical structures are now routinely evolved by computers. This automatic generation of innovation by creative evolutionary systems allows our designers to consider more solutions faster. These systems allow us to sidestep limitations of "conventional wisdom" and "design fixation." Creative evolutionary systems can even suggest entirely new methods and principles that we can then exploit in our own designs.

The list of applications of these approaches is long and increasing at a ferocious rate. But we are still only taking the first steps along this road. We are still learning how to enable evolutionary computation to provide us with this kind of assistance. There are as many problems yet to overcome as there are debates about the topic. The chapters in this book are our best answers so far.

Thankfully we have a very good mentor to inspire us and help us when we run into difficulties. Natural evolution is a very creative problem solver, and the solutions of nature are ever present to remind us just what evolution is capable of. Can *we* make a machine 3 millimeters long that is capable of flying under its own power? What about giving it sight, or the ability to keep itself functioning by converting chemicals into energy, or even the ability to make copies of itself? We simply do not have the know-how to achieve these marvels. But this is what nature has achieved in a creature as simple as a fly. Other examples abound. For example, would *we* have thought of turning flippers designed to propel fish through the water into arms and highly dexterous hands? Would *we* have thought of using hair to make the horn of a rhinoceros? Despite the constraints nature is faced with—only a small number of materials available, the fact that life must grow from a single cell and maintain itself—the creativity of nature is overwhelming. We do not have to wait for some science fiction scenario where aliens from other worlds give us superior technology. The technology of the natural world is already all around and within us—and it is vastly superior to our technology.

WHAT'S TO COME

This chapter gives an introduction to creative evolutionary systems. It is structured into three major sections: The first gives a general summary of evolutionary computation and the dominant evolutionary algorithms. The second provides definitions and reviews of the significant aspects of creative evolutionary systems, to place the contents and structure of the rest of the book into context. In the final section, we discuss the debate on creativity and evolution—is evolution really creative? But before we look at evolutionary computation, let's take a brief look at some of the other AI approaches used for creative applications.

AI AND CREATIVITY

Art, music, and designs have been emerging from computers for many years. Evolutionary systems are a promising technique for such enterprises, recently growing in favor. However, several other techniques have been used and explored with creative products in view. Artificial intelligence (AI) is the field within which much of this work takes place, and three main areas of that science are commonly involved in AI-based creativity research: *knowledge-based systems, grammars,* and *search.* Nonevolutionary creative systems are, by definition, not in the scope of this book, but it makes sense to provide the reader with the flavor of nonevolutionary approaches in this area. In fact, it seems reasonable to expect that some success in the creativity realm will emerge from hybridizations of evolutionary systems with other AI methods.

Knowledge-based systems (KBSs) come in very many forms, but their characteristic feature is their incorporation of expert knowledge in some domain, usually in the form of rules. Obviously, since a KBS needs experts' rules about a domain, the rules themselves need to be acquired from experts. According to many definitions, this would seem to rule out the possibility of a KBS being creative. However, a KBS is rescued by what will be a recurring theme in the chapters of this book. Results that seem usefully or pleasingly novel can arise from a "constrained exploration" within the parameters of the given knowledge. We see this often in musical KBSs that incorporate knowledge of the style of a particular composer, and which are then used to produce new compositions in that style. For example, Ebcioglu (1988) discusses a system that harmonizes chorales in the style of J. S. Bach, and Pachet, Ramalho, and Carrive (1996) discuss a system for real-time jazz improvisation. Both of these incorporate considerable domain knowledge about the styles of music in the domains of interest, and attempt to generate novelty within those constraints.

Grammars are an alternative way of representing knowledge in a particular domain. Grammars embody rules about languages. The grammar of English, for example, specifies that adjectives should come just before the nouns to which they are applied. The grammar of French begs to differ. With both, and indeed with all natural languages, violations of the rules tend to be allowed, and this lenience facilitates creativity. Grammars have played a part in the science of AI for various reasons, usually concerning the attempt to computationally understand natural languages, or translate between them. Their role in creativity stems from viewing designs or compositions as statements in a language. For example, pioneering work on this theme was done by Stiny and coauthors (Stiny and Mitchell 1978; Stiny 1980), who compiled a grammar for designing villas in the style of Palladio. Later work includes that of Knight (1980), who constructed a grammar for Hepplewhite chair backs; Koning and Eizenberg (1981), who present a grammar for generating house designs in the style of Frank Lloyd Wright prairie houses; and Flemming (1987), who presents a grammar for designing houses in the Queen Anne style.

Finally, search is a foundational concept in AI and refers to the long journey (made speedier via computation) through an immense space of possibilities in search of something suitable. We will discuss at length, in the rest of this introduction, how this is done by evolution. However, quite different approaches to search have long been a mainstay of AI research. The most explored search technique in AI is *heuristic search,* in which a solution (perhaps a design, perhaps a schedule, and so forth) is constructed gradually, bit by bit, with heuristics (rules of thumb) employed to decide how to choose each successive part. This is the technique employed by advanced Web spiders, for example, which explore the Web link by link in order to find useful new information on a particular topic. In this case, heuristics are used to decide which link to explore next. This is also the core technique used by the chess program Deeper Blue, which famously defeated the chess champion Garry Kasparov on May 11, 1997.[1] In this case, heuristics are used to decide which move to explore next; the search is of each reasonable possible move and its consequences, ultimately leading to a decision of which next move is most wise. The main difference between the evolutionary search systems we focus on in this book and classical AI heuristic search is a greater focus on heuristics in the latter. Heuristic searches tend to be quite fussy about the next move, concentrating on exploring areas that are sanctioned by the heuristics in use. Although this may seem limiting from the viewpoint of creativity, it remains the case that search-based systems can potentially aid the creative process by helping designers and artists see more of the space and possibilities than they otherwise would.

The Recommended Readings section at the end of this chapter provides several books for the interested reader who wishes to further explore either basic artificial intelligence or its role in creative applications.

EVOLUTIONARY COMPUTATION

Evolutionary computation is all about search. In computer science and in artificial intelligence, when we use a search algorithm, we define a computational problem in terms of a *search space,* which can be viewed as a massive collection of potential solutions to the problem. Any position, or point, in the search space defines a particular solution (Kanal and Cumar 1988), and the process of search can be viewed as the task of navigating that space. A commonly used term in this context is "optimization," which just means "finding the best." For example, just as you might browse through a bookshelf to find the book that best suits your needs, you might also browse through a search space to find the solution best suited to your specifications. This analogy is more appropriate than it may at first seem; one of the key ideas in search is that points close together in the space will also tend to be close in terms of quality. In your local bookshop, books on creative evolutionary computation are more likely to be close to books on evolutionary design than to books on optimization in general. So, if the first book you come upon during your bookshelf search happens to be *Evolutionary Design by Computers* (Bentley 1999a), you know that it will be worthwhile to look close by, perhaps even at the directly neighboring books on the shelf, to find a book about creative evolutionary systems. However, if the first book you come upon happens to be *New Ideas in Optimization* (Corne, Dorigo, and Glover 1999), you would do well to determine that looking at the directly neighboring books might be fruitless, and hence you might swing, perhaps, a foot either side, or a shelf up or down, in continuing your search.

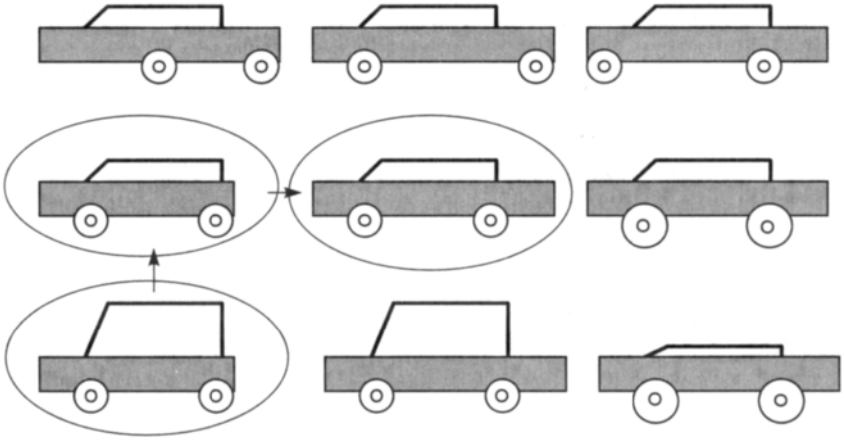
---

[1] In contrast, we note that evolutionary search, rather than classical AI heuristic search, is a key component in the similarly notable recent checkers player presented by Chellapilla and Fogel (1999).

More to the point, if during your search you happen to have found each of *Evolutionary Design by Computers, New Ideas in Optimization,* and *Evolutionary Algorithms in Engineering Applications* (Dasgupta and Michalewicz 1997), you may be able to do a kind of triangulation in this "bookshelf space." The kind of book you seek is likely to be closer to *Evolutionary Design by Computers (EDbC)* than to any of the others, but closer to *New Ideas in Optimization* than to *Evolutionary Algorithms in Engineering Applications.* If the latter two are found to be respectively to the left and right of *EDbC* on the same shelf, then you would more than likely be right to assume that the book you seek might be to the left of *EDbC;* hence, in a simple way, some knowledge about other solutions in the space has guided you helpfully (more likely than not) on your continuing quest.

Evolutionary search, and most other search methods, all make use somehow of previously visited solutions to help decide where to look next. Sometimes, the space includes a massive collection of terrible solutions that must be slowly waded through until you eventually find yourself in a decent neighborhood. In other cases, we may already have a good start. For example, we may already have a set of parameter values at hand that adequately define a car, as shown in Figure I.1. We might have parameters concerning distance between wheels, length of body, and so forth. The search shown in the figure moves through three car designs, where each move represents a change in one parameter. In general terms, this is precisely the sense in which neighboring solutions in a search space are "close"; that is, distance in the space is related to distance in terms of parameter settings.

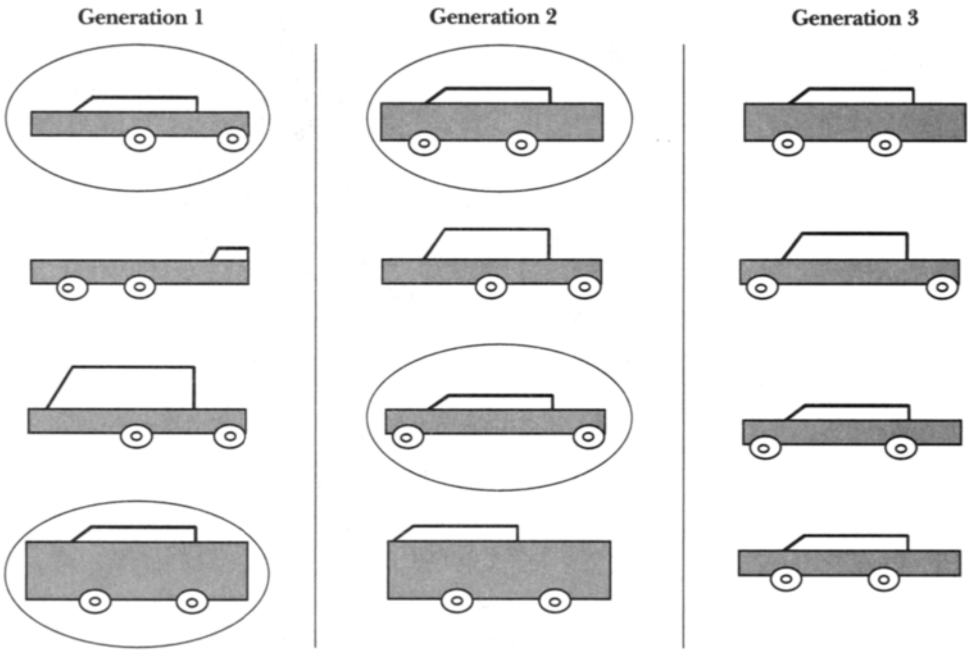
There are many types of search algorithms. Evolutionary search is a recent and rapidly growing subset and distinguishes itself from other methods in that the way it works is both inspired by and based upon the mechanism of evolution in nature. These algorithms typically use an analogy with natural evolution to perform search by *evolving* solutions to problems. Hence, instead of working with



I.1 FIGURE Searching for a solution in an example search space of car designs.

one solution at a time in the search space, these algorithms consider a large collection or *population* of solutions at once. In Figure I.1, we are actually illustrating a kind of search algorithm called *local search,* in which, at any point in time, we just have a single "current" solution, and we gradually update this with improvements if and when we find any better solutions nearby. In contrast, evolutionary algorithms work with populations of solutions. At any point in time, we have in mind (or, usually, in RAM) a population of potential solutions. We use the population as a whole (or at least we use more than one constituent of it) to help us determine where to go next in the space, as was illustrated in our bookshelf analogy.

Although evolutionary algorithms (EAs) do make computers evolve solutions, this evolution process is not explicitly specified in an EA; it is an *emergent property* of the algorithm. In fact, the computers are not instructed to evolve anything, and it is currently not possible for us to explicitly "program in" evolution—since we do not fully understand how evolution works. Instead, the computers are instructed to maintain populations of solutions, allow better solutions to "have children," and allow worse solutions to "die." The "child solutions" inherit their parents' characteristics with some small random variation, and then the better of these solutions are allowed to have children themselves, while the worse ones die, and so on. This simple procedure causes evolution to occur, and after a number of generations the computer will have evolved solutions that are substantially better compared to their long-dead ancestors at the start. In Figure I.2, for example, we can see three snapshots (generations) of an evolution

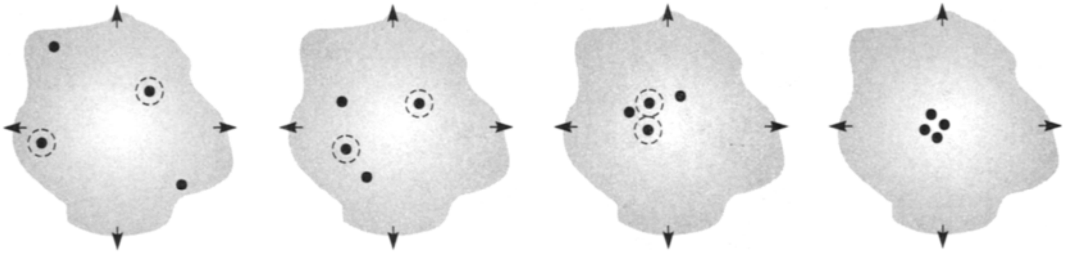


I.2 FIGURE Three generations of evolving car designs using a population size of four. Parents of the next generation are circled.

process, with time moving toward the right. The circled solutions are those that were used as "parents" in generating the succeeding generation.

By considering the search space itself, it is possible to get an idea of how evolution manages to find good solutions. Figure I.3 depicts the search space for the example shown in Figure I.2. It should be clear that evolution searches the space in parallel (in the example, it considers four car designs at a time). It should also be clear that evolution quickly "homes in" on the best area of the search space, resulting in some good designs after only four generations.

All EAs require some form of guidance to direct evolution toward better areas of the search space. In our bookshelf analogy, the guidance used was an intuitive measure of distance: in seeking a book about creative evolutionary systems we felt, for example, that a book about evolutionary design was closer to our goal than a book about telecommunications. Implicitly, we had a good look at each book encountered, and evaluated it in terms of its fitness for our requirements. The better the evaluation score, the closer we felt it to be to our goal. In general, EAs receive guidance in just the same way, by means of *evaluating* every solution in the population, to determine its *fitness*. The fitness of a solution is a score

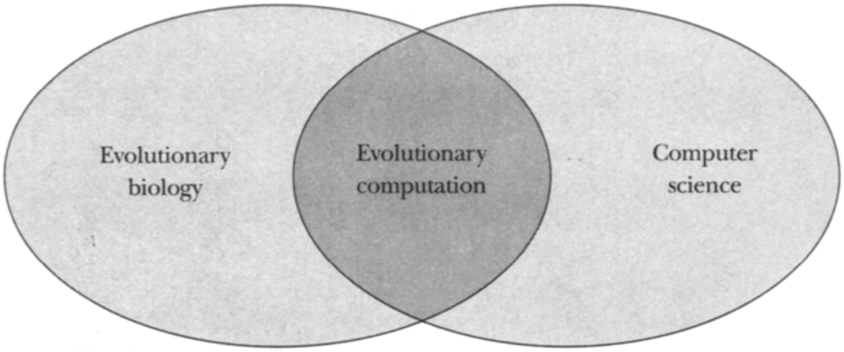


I.3 FIGURE The location of the evolving car designs in the space of car designs, for each generation. Better solutions are found in the center of this example space.

based on how well the solution fulfils the problem objective, calculated by a *fitness function.* Typically, fitness values are positive real numbers, where a fitness of zero is a perfect score. EAs are then used to minimize the fitnesses of solutions, by allowing the fitter solutions to have more children than less fit solutions. In the car example, the problem objective might be to find a car design that has adequate legroom, suitably placed wheels (to ensure stability), and so on. The fitness function would take a solution as input and return a fitness value based on how well the solution satisfies these objectives.

Fitness values are often plotted in search spaces, giving mountainous *fitness landscapes,* where a high peak corresponds to solutions in that part of the search space that have optimal fitnesses. If the problem has many separate optima (i.e., if the fitness function is *multimodal*), finding a globally optimal solution (the top of the highest mountain) in the landscape can be extremely difficult.

There are four main families of evolutionary algorithm in use today, three of which were independently developed more than 30 years ago. These algorithms are the *genetic algorithm* (GA), created by John Holland (1973, 1975) and made famous by David Goldberg (1989); *evolutionary programming* (EP), created by Lawrence Fogel (1963) and developed further by his son David Fogel (1992a); and *evolution strategies* (ES), created by Ingo Rechenberg (1973) and today strongly promoted by Thomas Bäck (1996). The fourth major evolutionary algorithm is a more recent and very popular variation of the genetic algorithm by John Koza (1992), known as *genetic programming* (GP). The field of evolutionary computation has grown up around these techniques, with its roots still firmly in evolutionary biology and computer science (see Figure I.4). Today researchers examine every conceivable aspect of EAs, often using knowledge of evolution from evolutionary biology in their algorithms, and more recently, using EAs to help biologists learn about evolution (Dawkins 1986).



1.4 FIGURE Evolutionary computation has its roots in computer science and evolutionary biology.

Evolution-based algorithms have been found to be some of the most flexible, efficient, and robust of all search algorithms known to computer science (Goldberg 1989). Because of these properties, these methods are now becoming widely used to solve a broad range of different problems (Holland 1992; Mitchell 1996).

In the following sections we briefly summarize the four dominant types of EA, and then we provide an overview of a general architecture for EAs, to show how these separate techniques follow a common evolutionary paradigm. But first, we look back at some early work that sowed the seeds for this field.

Seminal Computer Evolution[2]

In the early 1950s, the well-known statistician George E. P. Box came up with an innovative idea first written up in an internal report for Imperial Chemical Industries Ltd., for improving productivity in a chemical process plant. Calling the idea "evolutionary operation," Box proposed a system whereby plant control settings (such as temperature and percentage concentration of reactants in a chemical process) could be deliberately and systematically varied during plant operation, and the results (in terms of the percentages of different impurities of the chemical being manufactured) could be recorded. Control settings for further batches could then be set according to an analysis of the results found so far.

---

[2] We are grateful to David Fogel for pointing us toward information on this topic.

Despite going severely against the grain—production managers are notoriously reluctant to alter control settings for experimental purposes when things are apparently going quite well without such fiddling—Box and coworkers' ideas were good enough, and convincing enough in terms of potential increased productivity, to lead to "evolutionary operation" being put into practice in several chemical firms in the late 1950s and early 1960s.

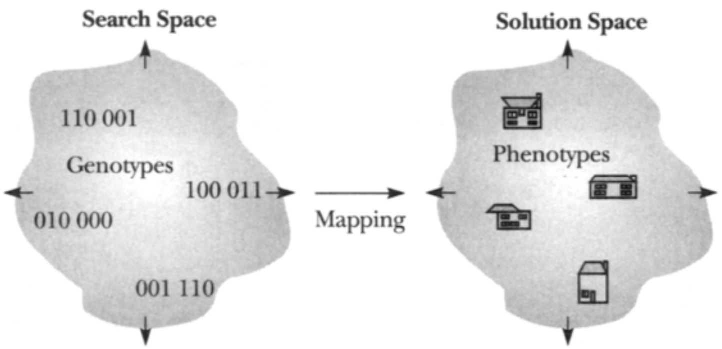
Evolutionary operation first appeared in a generally accessible publication as Box (1957). In use, it was always what we would now call a "collaborative evolutionary system"; that is, it was not entirely automatic—a human, or a committee of humans, needed to be involved in order to decide how to set the controls for the next generation of trials. The human's role in Box's method differs somewhat from the human's role in modern collaborative evolutionary systems.

In 1956, George Friedman, a master's student at the University of California, Los Angeles, designed what he called a "selective feedback computer." This was a design for a robotic machine that, using Darwin's principle of natural selection, would evolve circuit designs. This machine was never implemented, but the ideas in Friedman's thesis were profoundly perceptive in the way that the potential of evolutionary style search was demonstrated. Around this time, there were other independent developments of similar ideas. For example, Friedberg (1958) attempted to (effectively) evolve machine code to perform simple arithmetic operations.

The definitive source for information on such pioneering early studies is the aptly named "Fossil Record" book, edited by Fogel (1998). We refer the interested reader there for further meanderings in evolutionary computation's primordial soup. In the following sections, however, we present the current state of the field of evolutionary computation, whose roots tend to be in the 1970s and 1980s, when a range of researchers, helped (unlike their 1950s and 1960s precursors) by the availability of decent programming languages and fast machines, began to find that evolution-based ideas in computer science could have profound and real practical value.

Genetic Algorithms

The genetic algorithm is perhaps the most well known and popularized of all evolution-based search algorithms, although it is fair to say that this is partly a result of the term "genetic algorithm" being often used to denote each of the four main families of methods (i.e., it is often used in the way we use the term "evolutionary algorithm"). GAs were developed by John Holland in an attempt to explain the adaptive processes of natural systems and to design artificial systems



I.5 FIGURE Mapping genotypes in the search space to phenotypes in the solution space.

based upon these natural systems (Holland 1973, 1975). Although not the first algorithm to use principles of natural selection and genetics within a search process, and although not necessarily the "best" approach to take in solving a particular given problem, the genetic algorithm is today, probably, the most widely used of the four main kinds of EA. More experimental and theoretical analyses have been made on the workings of the GA than on the other EAs. Also, the genetic algorithm (and enhanced versions of it) resembles natural evolution more closely than do most other methods; but, again, the extent to which this matters in applications will vary greatly.

Having become widely used for a broad range of optimization problems in the last 15 years (Holland 1992), the GA has been described as a "search algorithm with some of the innovative flair of human search" (Goldberg 1989). GAs are also very forgiving algorithms—even if they are badly implemented, or poorly applied, they will often still produce acceptable results (Davis 1991). GAs are today renowned for their ability to tackle a huge variety of optimization problems and for their consistent ability, given at least *some* thought into a suitable setup for the application in hand, to provide excellent results; that is, they are *robust* (Holland 1975; Goldberg 1989; Davis 1991; Fogel 1994).

It is fruitful to view a GA as making use of two separate spaces: the *search* space and the *solution space.* The search space is now a space of *coded* solutions to the problem, and the solution space is the space of actual solutions. Coded solutions, or *genotypes,* must be mapped onto actual solutions, or *phenotypes,* before the quality or fitness of each solution can be evaluated (see Figure I.5).

GAs maintain a population of *individuals,* where each individual consists of a genotype and its corresponding phenotype. Phenotypes usually consist of collections of parameters (in our car example, as we saw, such parameters might

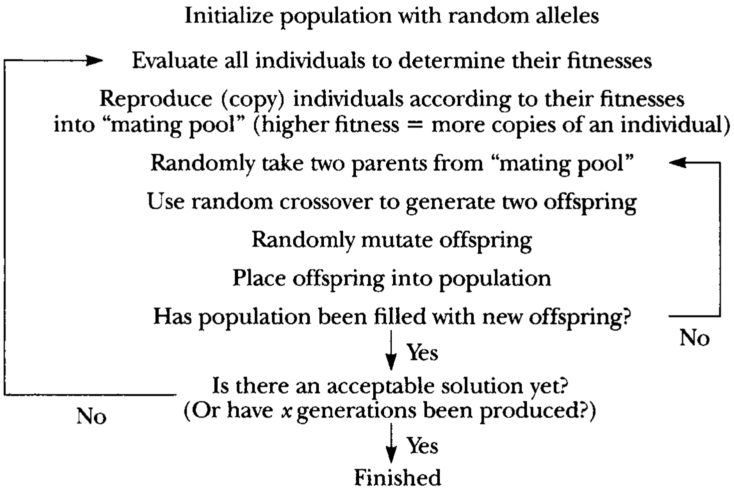
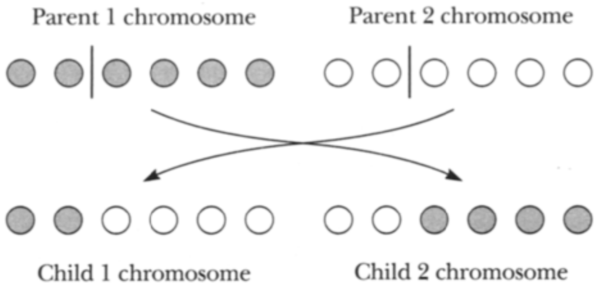

Initialize population with random alleles
Evaluate all individuals to determine their fitnesses
Reproduce (copy) individuals according to their fitnesses into "mating pool" (higher fitness = more copies of an individual)
Randomly take two parents from "mating pool"
Use random crossover to generate two offspring
Randomly mutate offspring
Place offspring into population
Has population been filled with new offspring?
No
Yes
Is there an acceptable solution yet? (Or have x generations been produced?)
No
Yes
Finished

FIGURE I.6 The simple genetic algorithm.

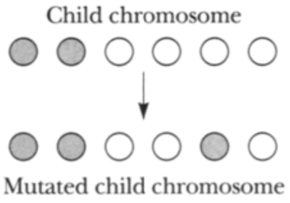
define the distance between wheels, the length of the body, and so on). Genotypes consist of coded versions of these parameters. A coded parameter is normally referred to as a *gene,* with the values a gene can take being known as *alleles.* A collection of genes in one genotype is often held internally as a string and is known as a *chromosome.*

The simplest form of GA, the *canonical* or *simple GA,* is summarized in Figure I.6. This algorithm works as follows: The genotype of every individual in the population is initialized with random alleles. The main loop of the algorithm then begins, with the corresponding phenotype of every individual in the population being evaluated and given a fitness value according to how well it fulfils the problem objective or fitness function. These scores are then used to determine how many copies of each individual are placed into a temporary area, often termed the "mating pool" (i.e., the higher the fitness, the more copies that are made of an individual).

Two parents are then randomly picked from this pool. Offspring are generated by the use of the crossover operator, which randomly allocates genes from each parent's genotype to each offspring's genotype. For example, given two parents: 'ABCDEF' and 'abcdef', and a random crossover point of, say, 2, the two offspring generated by the simple GA would be 'ABcdef' and 'abCDEF' (see Figure I.7). (Crossover is used about 70% of the time to generate offspring; for the remaining 30% offspring are simply clones of their parents.) Mutation is then occasionally applied (with a low probability) to offspring. When it is used to mutate



I.7 FIGURE The behavior of the crossover operator. The vertical line shows the position of the random crossover point.



I.8 FIGURE The behavior of the mutation operator.

an individual, typically a single allele is changed randomly. For example, an individual '110000' might be mutated into '110010', as illustrated in Figure I.8.

Using crossover and mutation, offspring are generated until they fill the population (all parents are discarded). This entire process of evaluation and reproduction then continues until either a satisfactory solution emerges or the GA has run for a specified maximum number of generations (Holland 1975; Goldberg 1989; Davis 1991).

The randomness involved in the genetic operators can give the illusion that the GA, and other EAs, are nothing more than parallel random search algorithms, but this is far from the case. Evolutionary search has a random element to its exploration of the search space, but the search is unquestionably *directed* by the "survival of the fittest" principle. In the simple example GA algorithm just described, this principle comes into play when we decide which chromosomes can join the mating pool and hence be parents for the next generation. This decision process is called *selection,* and as long as the decision is made in such a way that better fitness gives more chance of being selected, the survival of the fittest principle is in operation, and there is *selection pressure* toward areas in the search space that contain better solutions.

Unless the genetic operators are very badly designed, an EA will always "home in" on these areas. The same is true, to some extent, of *local search* methods, which navigate the space one solution at a time, in the way we saw illustrated by Figure I.1. It is easy to imagine, however, that such a process might soon get stuck at a so-called *local optimum*—a solution that is far from best, but whose immediate neighbors in the space are all even worse. Part of the power and success of EAs lies in the fact that the insistence on maintaining a population of solutions is very helpful in avoiding such dead ends (Goldberg 1989).

However, the simple GA is just that—very simple and a little naïve. This type of GA is generally not capable of solving difficult search problems, but is favored by those that try to theoretically analyze and predict the behavior of genetic algorithms. In practice, typical applied GAs are usually considerably more advanced. The basic principles remain—a population of solutions, evolving according to the survival of the fittest principle, and new candidate solutions are produced by mutation and/or crossover operators—but the details are usually quite different. Common features include more advanced selection mechanisms, more sophisticated genetic operators, incorporation (hybridization) with other algorithms, ability to detect when evolution ceases, and overlapping populations or elitism (where some fit individuals can survive for more than one generation) (Davis 1991). Many of the typical improvements over the simple GA are in the direction of increased similarity to natural evolution: for example, the use of spatially structured populations, slower gradual change in the population membership, inclusion of an "age" element in the fitness calculation, genotype-to-phenotype mappings that involve something akin to "development," and so forth. In line with this improved analogy with nature, the term *reproduction* is normally used as it is in biology to refer to the entire process of generating new offspring, encompassing the crossover and mutation operators. (This is in contrast to the somewhat confusing use of the word "reproduction" in some GA literature, where it refers to an explicit copying stage within the simple GA.)

Advanced Genetic Algorithms

It is in the nature of complex problem solving that we can never guarantee (except for trivial problems that we can solve easily anyway) that an evolutionary algorithm or any other algorithm will find an optimal solution in reasonable time. However, we would at least expect our EA to have a good stab at the problem and attempt to exploit fully the information in the population and the power of its operators. A common problem arises when this doesn't happen, and we have a situation termed *premature convergence*. This is where the population converges early onto nonoptimal solutions, and further evolution seems unable to push toward anything better (Davis 1991). Even if we can avoid premature convergence (so that the algorithm at least exploits all the time available to it), we remain far from guaranteed to converge on anything like the best solutions possible. Highly deceptive problems can be defined, for example, that will cunningly mislead the course of evolution. In addition, noisy functions (Goldberg et al. 1992) and the optimization of multiple criteria within can cause other difficulties (Fonseca and Fleming 1995). In an attempt to address such problems, new, advanced types of GA are being developed. These include the following:

- *Steady-state GAs,* where offspring are generated one at a time and replace existing individuals in the population according to fitness or similarity. Convergence is slower, but very fit solutions are not lost (Syswerda 1989).
- *Parallel GAs,* where multiple processors are used in parallel to run the GA (Adeli and Cheng 1994; Levine 1994).
- *Distributed GAs,* where multiple populations are separately evolved with few interactions between them (Whitley and Starkweather 1990).
- *GAs with niching and speciation,* where the population within the GA is segregated into separate "species" (Horn 1993; Horn and Nafpliotis 1993; Horn et al. 1994).
- *Messy GAs (mGAs),* which use a number of "exotic" techniques such as variable-length chromosomes and a two-stage evolution process (Deb 1991; Deb and Goldberg 1991).
- *Interactive GAs (IGAs),* or *collaborative GAs,* where user interaction takes over some or all of the role of the fitness function.
- *Multiobjective GAs (MOGAs),* which allow multiple objectives to be optimized with GAs (Schaffer 1985; Srinivas and Deb 1995; Bentley and Wakefield 1997c; Zitzler and Thiele 1999; Knowles and Corne 2000).
- *Hybrid GAs (hGAs),* for example, memetic algorithms, where GAs are combined with local search algorithms (Radcliffe and Surrey 1994a; Moscato 1999).
- *Structured GAs (sGAs),* which allow parts of chromosomes to be switched on and off using evolvable "control genes" (Dasgupta and McGregor 1992; Parmee and Denham 1994).
- *GAs with diploidy and dominance,* which can improve variation and diversity in addition to performance (Smith and Goldberg 1992).
- *Mutation-driven GAs,* such as Harvey's SAGA (Harvey 1992), which uses converged populations modified primarily by mutation to allow the constant "incremental evolution" of new solutions to varying fitness functions.

- *GAs with "genetic engineering,"* which identify beneficial genetic material during evolution and prevent its disruption by the genetic operators (Gero and Kazakov 1996).
- *Injection Island GAs (IIGAs),* which evolve a number of separate populations ("islands") with representations and fitness functions of different accuracy and occasionally "inject" good solutions from one island into another (Eby et al. 1997).

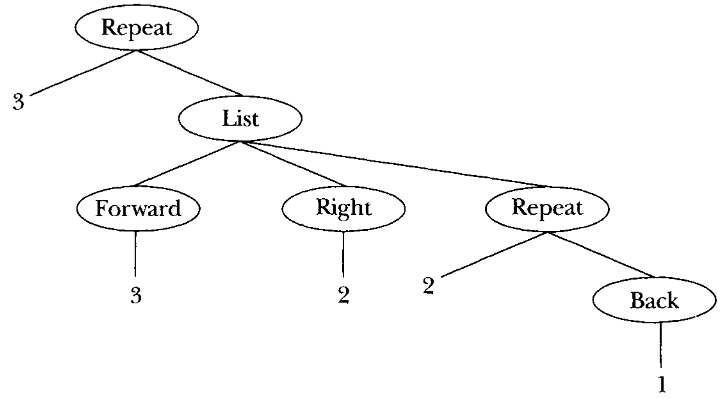
This is by no means a complete list. Some of these advanced types of GA are described further in the chapters of this book. Recommended readings for GAs can be found at the end of this chapter.

Genetic Programming

Genetic programming (GP) is a specialized form of genetic algorithm, which manipulates a very specific type of chromosome using genetic operators that are carefully crafted to the task at hand. The kinds of chromosome manipulated in GP are *programs,* which can usually be represented as treelike structures. Seeds were sown for this type of idea by Cramer (1985), but GP was developed and popularized by Koza (1992), who managed to show convincingly that this idea has great potential. Practitioners of GP are beginning to move away from the original application of evolving computer programs, with GP now being applied in alternative areas, creative evolutionary applications among them. John Koza describes one such application in Chapter 10 of this book.

The underlying algorithmic structure of GP is essentially the same procedure as already described for GAs. Where GP distinguishes itself is in the nature of the chromosome and operators. Rather than the fixed-length string-based representation common in GAs (and other types of EA), GP evolves variable-sized hierarchical tree structures that can be interpreted as computer programs or functions. You should be able to see how the chromosome in Figure I.9 might represent a simple program to control a robot. In pseudocode, this program might be

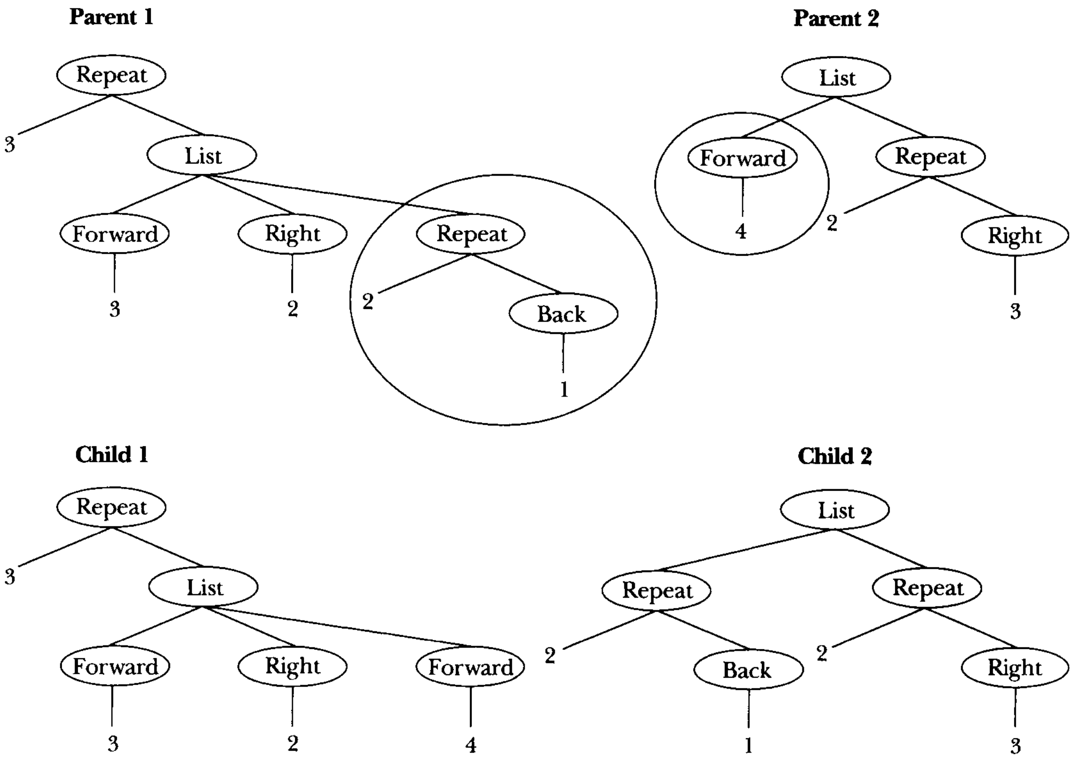
```
repeat 3 times
   { move 3 steps forward
     move 2 steps to the right
     repeat 2 times
        { move back 1 step
```



I.9 FIGURE A simple computer program defined by GP's hierarchical representation.

Clearly, crossover operators and mutation operators of the kind illustrated in Figures I.7 and I.8 are not suitable here. Koza's approach, and what is now standard in GP, is illustrated in Figure I.10. Randomly chosen branches of two parent trees are simply interchanged to produce a pair of child trees that will typically be quite different from each other and from either parent. GP's mutation operator is also specialized in the same sort of way. This operator picks a random subtree and replaces it entirely with a new randomly generated subtree. In Figure I.10, for example, Child 1 could have been generated as a mutant of Parent 1, where the highlighted subtree in Parent 1 was the one deleted, and the newly generated random subtree in its place happened to be the one shown. Actually, partly owing to the ability of crossover to handle the role of mutation in GP, mutation in GP is sometimes considered unnecessary (Koza 1992).

The typical applications addressed by GP bring along other specialized baggage, most notably in connection with the evaluation of solutions. In a typical EA application, as we have described, we usually have a routine at hand that we can use to assign a fitness score to a candidate solution, and the calculation is often relatively straightforward. In GP, however, a solution is a program, and we must *run* the program, usually several times, to get an idea of how good it is at meeting the requirements. Alternatively (and this is also sometimes the case with other EAs), a single run of the program is needed to generate or "grow" a solution, and then we can apply a straightforward fitness function to it. Normally, when we are trying to evolve a program, a series of input values and desired output values are provided, and the fitness of the program is based on how closely actual output values match the desired output values, for each set of input values.


Parent 1
Repeat
3
List
Forward
3
Right
2
Repeat
2
Back
1
Parent 2
List
Forward
4
Repeat
2
Right
3
Child 1
Repeat
3
List
Forward
3
Right
2
Forward
4
Child 2
List
Repeat
2
Back
1
Repeat
2
Right
3

1.10 FIGURE The behavior of the crossover operator in GP. The highlighted subtrees in the parents have been exchanged to produce the children.

Programs evolved by GP, or indeed created by humans, may often contain redundant or unused code; such code is often called *junk* or *introns*. For example, the following simple robot control program contains two types of junk. Can you tell where?

```
repeat 5 times
   { move 3 steps forward
     move 2 steps to the right
     move 1 step forward
     move 1 step backward
     repeat 2 times
          { move back 1 step
          }
     repeat 0 times
```

```
            { move 10 steps forward
            }
    }
```

Such junk-filled programs are common in standard GP, with most solutions steadily increasing in size as evolution progresses. This tendency is known as *bloat* (Langdon and Poli 1997), and GP practitioners tend to limit its effect by adding a penalty to the fitness of any solutions that become oversized.

GP-style programs may also, of course, contain conditional statements, such as "IF A THEN B ELSE C". When such programs are evolved, the effects are interestingly similar to the concept of *dominance* in natural genetics. For example, in the following conditional statement, the action `move-right(3)` can be considered dominant, and the action `move-ahead(2)` recessive, since we can expect the `move-right(3)` action to be the one employed about 75% of the time:

```
IF (bearing = NORTH) THEN move-ahead(2) ELSE move-right(3)
```

In other EAs, however, simulation of dominance and recessive characteristics requires sophisticated extensions to the method. For example, a GA might use multiploid chromosomes (a single candidate solution, or *genome*, being composed of two or more distinct chromosomes) where the *expressed* value of a gene depends somehow on the values of the various copies of that gene in the genome. Conditional statements in GP also resemble the operons and regulons found in our own DNA, which seem to be used to switch on and off other genes during the development of the organism (Paton 1994).

Advanced GP

GP implementations often employ further specialized operators, including *permutation,* which swaps two characters in a tree; *editing,* which allows the optimization and reduction of long S-expressions; and *encapsulation,* which allows a subtree to be converted into a single node, preserving it from disruption by crossover or mutation.

A well-known and now commonly used advance, which can be seen as a more advanced version of encapsulation, is the evolution of *automatically defined functions* (ADFs). Essentially, after a while, certain subtrees that seem to be "good" become treated as units. Typically, the GP system is given a predetermined number of possibilities for the nodes and terminals that can appear in trees. When a subtree becomes a unit, or an ADF, it is added to this set of possibilities. This saves having to re-evolve it from scratch each time, and also reduces the danger of disruption of good units by crossover or mutation. ADFs have been shown to considerably enhance the performance of GP (Koza 1992).

Researchers are exploring other styles of function creation. For example, Yu evolves anonymous functions known as λ-abstractions, which are reused through recursion (Yu and Clack 1998a, 1998b). Current research also investigates the automatic creation of iterations (ADIs), loops (ADLs), recursions (ADRs), and various types of memory stores (ADS) (Koza et al. 1999).

Finally, it is worth noting that typical operators employed in GP have problems with excessive disruption (O'Reilly and Oppacher 1995). In standard EAs with standard styles of chromosome and standard operators, a useful proportion of offspring generated using crossover or mutation from fit parent solutions will be at least as fit as their parents. But, in GP, the proportion of such successes is very much reduced (which is one reason why GP implementations tend to employ significantly larger populations and run for significantly longer than other EAs). Such observations have led to the development of new crossover operators, designed to minimize the disruption of good solutions. The typical strategy employed by such operators is to restrict the use of crossover to the recombination of similarly structured parents (Poli and Langdon 1997a). See the recommended reading section at the end of this chapter for books on GP.

Evolution Strategies

Evolution strategies (or *evolutionsstrategie*) were pioneered in Germany in the 1960s by Bienert, Rechenberg, and Schwefel (Bäck 1996). The first applications of this technique were optimizing the shape of a bent pipe, drag minimization of a joint plate, and structure optimization of nozzles. Somewhat akin to Box's "evolutionary operation" scheme (discussed earlier), the processors involved in their original implementation were biochemical, rather than electromagnetic—physical designs were built, tested, and mutated by manually altering joint positions or adding and removing segments.

Fully automated evolution strategies began with Schwefel (1965) and continued with Rechenberg (1973). This work concentrated on the simplest form of ES, known as the *two-membered* ES. Using the terminology we have introduced in describing the GA, we can describe the simplest form of ES as an elitist EA with a population size of 1. Naturally, this means that there is no use for a crossover operator, and the selection process is somewhat simplified. The entire algorithm works as follows: Beginning with the generation and evaluation of an initial random solution, which will be the first "parent" solution, a child solution is generated by randomly mutating the parent, and the child is then evaluated. If the child turns out better than (or better fit as) the parent, then it now becomes the parent and the old parent is discarded. Otherwise, the child is discarded. Now, we simply iterate the process. This overall scheme is known as a (1+1)-ES. The 1s stand for the number of parents and the number of children, respectively. We'll see later how things work when one or both of these numbers is other than 1.

There are two other prominent differences between ESs and canonical GAs. Early GAs (but this is less common nowadays) tended to use binary chromosomes. That is, a solution to a problem was somehow represented as a string of zeroes and ones. Even if the solution was meant to be a list of parameter values, the chromosome represented these as binary numbers, and mutation happened at the bit level (e.g., rather than adding or subtracting an arbitrary small amount to a parameter, we would directly flip one of the bits in its binary representation). Early (and most modern) ESs tend to be applied to problems involving strings of real-valued parameters, and the chromosome is a direct list of the parameters themselves, rather than a binary version of them. Mutation in an ES involves making small additions or subtractions (generally, *deviations*) to parameters. Such mutations are guided, in the simplest ES, by a constant standard deviation parameter. That is, a random number generator is primed to produce random deviations according to a Gaussian distribution with a mean of zero and a fixed standard deviation.

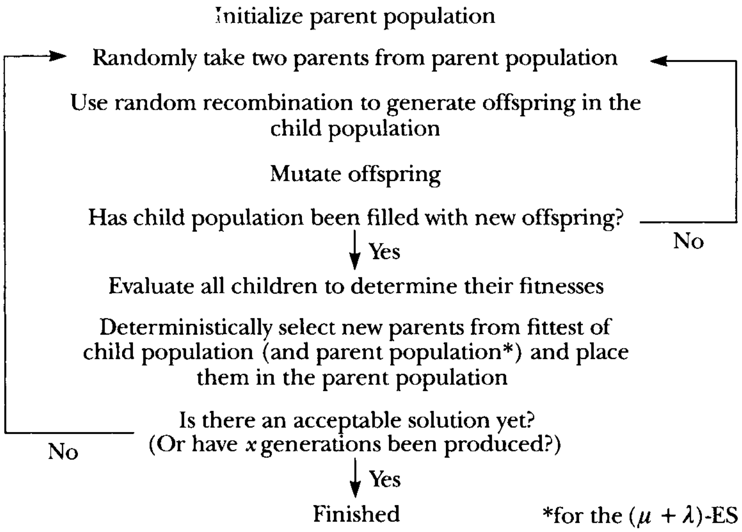
Problems with these early ESs led to certain solutions being developed, which now form the main characteristic difference between a typical modern ES and other EAs. This difference is in the inclusion of *strategy parameters* in a chromosome. Basically, with these parameters, the typical deviations applied to a gene by mutation can now vary over the course of the algorithm's run and can also vary with different genes. Imagine, for example, a chromosome consisting of five real-valued genes. An example chromosome in a standard EA might be

1.25 3.81 4.12 −7.04 0.83

In a modern ES, however, we will typically encode this solution as 10 "genes," for example,

1.25 3.81 4.12 −7.04 0.83 0.1 0.02 0.03 0.34 0.001

The extra collection of genes encode parameters guiding the sizes of the deviations that will be applied to the original genes. In this case, the first five genes correspond directly to parameter values, but the ith gene, where $i > 5$, encodes the variance of the deviations that will be applied to the $(i - 5)$th gene. So, in future generations, referring to the example chromosome above, rather larger deviations will be applied to the first gene than to the second.


Initialize parent population
Randomly take two parents from parent population
Use random recombination to generate offspring in the child population
Mutate offspring
Has child population been filled with new offspring?
No
Yes
Evaluate all children to determine their fitnesses
Deterministically select new parents from fittest of child population (and parent population*) and place them in the parent population
Is there an acceptable solution yet? (Or have x generations been produced?)
No
Yes
Finished
*for the (μ + λ)-ES

I.11 FIGURE **The evolution strategy.**

More population-oriented ESs soon followed, and the current state of the art (Bäck 1996) can be captured by the general algorithmic scheme in Figure I.11. The ES is initialized with a population of random solutions, or from a population of solutions mutated from a single solution provided by the user. Parents are chosen randomly from the "parent population," and a number of random recombination operators are used to generate child solutions, which are placed in the "child population." These operators may recombine the values within two parents like the crossover operator of GAs, or they may use a parent for every decision variable in the solution. New solutions are then mutated using *strategy parameters* within each solution, and a special scheme is employed to also slightly mutate the strategy parameters themselves. Once mutation and recombination has generated enough new individuals to fill the child population, these individuals are evaluated to obtain fitness measures, as described earlier for GAs. The fittest offspring are then deterministically selected to become parents, and these are placed into the parent population. With the parent population filled with (mostly) new individuals, the ES randomly picks parents from this population to generate new child solutions, which are then evaluated, and so on. Evolution terminates after a predefined number of generations, or when a solution of sufficient quality has been generated.

Mutation plays an important role in ES and is regarded as the primary search operator. Unlike the entirely random mutation common to simple GAs and GP, the mutation of ES follows a *normal distribution;* the simple strategy parameters we have discussed so far are used to specify a particular normal distribution for each gene—this distribution always has mean zero, and its standard deviation is given by the strategy parameter itself.

Advanced ES

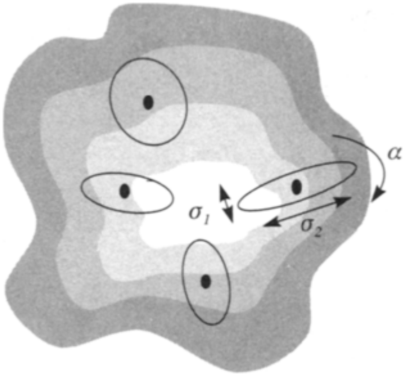
In advanced ES, another kind of strategy parameter is also employed: a rotation angle α. Depending on the number of genes that specify a solution, a suitable number of angle strategy parameters, in addition to the deviation parameters, are used to influence the correlation of mutations of one gene with another. The result, as illustrated in Figure I.12 for a simple two-gene case, is that the strategy parameters for a solution define a kind of "probability hyperellipsoid" that characterizes the effect of mutations on that solution. The strategy parameters thereby influence the direction of search in the search space taken by mutation, and by adapting the strategy parameters over time, the ES is able to use mutation to follow the contours of the search space and quickly find good solutions.

By using strategy parameters in this way—in particular, having them encoded as part of a solution and themselves subject to genetic operators—the ES employs a technique known in the general EA world as *self-adaptation.*

By the early 1980s, the current state-of-the-art evolutionary strategies had been developed (Bäck 1996), known as the $(\mu + \lambda)$-ES and (μ, λ)-ES (where μ is the number of parents and λ is the number of offspring). These new types of ES now strongly resemble the genetic algorithm, by maintaining populations of individuals, selecting the fittest individuals, and using recombination and mutation operators to generate new individuals from these fit solutions.

There are some significant differences between ES and GAs, however. For example, although ES maintains populations of solutions, it separates the parent individuals from the child individuals. In addition, as mentioned above, ES does not manipulate coded solutions like GAs. Instead, like GP, the decision variables of the problem are manipulated directly by the operators. Also unlike the probabilistic selection of GAs and GP, ES selects its parent solutions deterministically.

The $(\mu + \lambda)$-ES picks the best μ individuals from both child and parent populations. The (μ, λ)-ES picks the best μ individuals from just the child population. In order to ensure that a selection pressure is generated, the number of parents, μ, must always be smaller than the number of offspring, λ. Bäck (1996)



I.12 FIGURE

Mutation hyperellipsoids defining equal probability density around each solution. In this example, each solution has two σ parameters and one α parameter to guide mutation. Note how mutations toward the better area of the search space are encouraged.

recommends the use of the (μ, λ)-ES with a parent:offspring ratio of 1:7. See the recommended reading section at the end of this chapter for books on ES.

Evolutionary Programming

Evolutionary programming was developed by Lawrence Fogel in the early 1960s. Its algorithmic form closely resembles that of evolutionary strategies, although evolutionary programming was developed independently (and earlier). The original Evolutionary Programming algorithms were applied to the evolution of state transition tables for finite-state machines (FSMs). In fact, evolutionary programming was put forward as a procedure to generate intelligent machine behavior in the early days of artificial intelligence. With intelligent behavior characterized by the ability to predict one's environment and to provide a suitable response in order to achieve a given goal, FSMs were viewed as flexible general architectures for specifying such behavior, and hence Fogel developed the idea of evolutionary programming specifically for the purpose of evolving FSMs.

In Fogel, Owens, and Walsh (1966), for example, a sequence of inputs, such as 010001, is presented to an FSM and plays the role of the "environment" within which the FSM is expected to behave intelligently. The FSM's task is to predict what the next symbol will be. A single population of solutions was maintained, each specifying a state transition table for an FSM, and reproduction used mutation alone.

Much research was carried out along these lines at New Mexico State University during the 1970s, but evolutionary programming was either misunderstood or overlooked for several years, until David Fogel revisited and redeveloped it in the late 1980s. There is now a thriving and growing community of EP researchers, mainly using D. Fogel's extensions to L. Fogel's original ideas and applying EP successfully to a vast range of practical problems.

Advanced EP

David Fogel's extensions of the original EP rendered it applicable to continuous parameter optimization (fixed-length real-valued vectors) and other representations, such as permutations for the TSP, and also introduced a self-adaptation scheme. Modern EP thus shares much in common with modern ES, but there are various technical differences.

There are three main types of EP in common use: *standard EP, meta EP,*[3] and *Rmeta EP.* These three types differ by the level of self-adaptation employed. Standard EP uses no self-adaptation, meta EP incorporates mutation variance parameters in individuals to allow self-adaptation, and Rmeta EP incorporates mutation variance and covariance parameters into individuals to permit more precise self-adaptation (Fogel 1995). Figure I.13 shows the working of a general evolutionary program.

Like ES, implementations of EP tend to operate on the decision variables of the problem directly (i.e., the search space is the same as the solution space). During initialization, these variables are given random values, often with a uniform sampling of values between predefined ranges. These solutions are then evaluated to obtain the fitness values (which in EP may involve some form of scaling or the addition of a random perturbation). Next, parents are picked using a specialized kind of *tournament selection.* Tournament selection is commonly used in GAs and involves a parameter called the tournament size t. When $t = 3$, for example, selection of a single parent is done by randomly choosing three chromosomes from the population and simply taking the fittest of these (breaking ties randomly) to be the chosen parent. Obviously, two applications of this are needed to get two parents for a crossover operation. The specialized tournament selection method commonly used in EP, however, adds some sophistication to this. In Chellapilla's version (1998), it works as follows. First, a pool of child chromosomes are produced via applying mutation operators to the population. Each chromosome in the combined pool (the parents and their mutant children) then participates in a tournament with q other randomly chosen individuals. Thereby, each chromosome in the combined pool records a number of "wins"

---

[3] Meta EP is now the standard form of EP in use today and is usually called simply "EP."

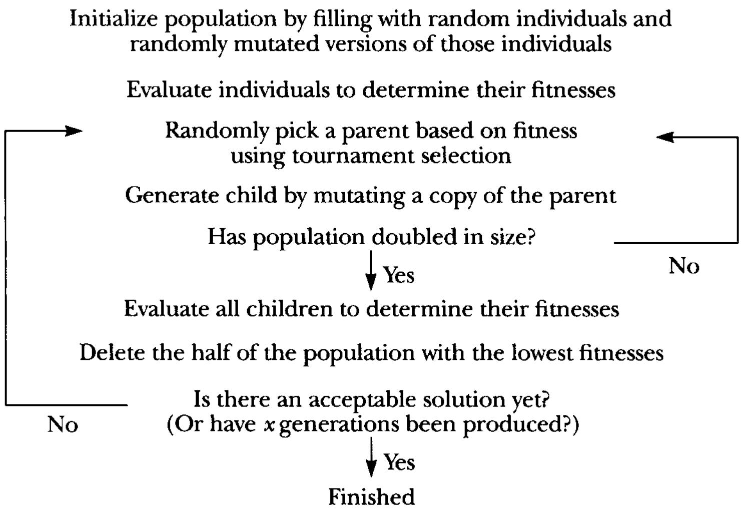

Initialize population by filling with random individuals and randomly mutated versions of those individuals
Evaluate individuals to determine their fitnesses
Randomly pick a parent based on fitness using tournament selection
Generate child by mutating a copy of the parent
Has population doubled in size?
Yes
No
Evaluate all children to determine their fitnesses
Delete the half of the population with the lowest fitnesses
Is there an acceptable solution yet? (Or have x generations been produced?)
No
Yes
Finished

FIGURE 1.13 The evolutionary programming algorithm.

(the number of the q competitors whose fitness it equalled or bettered). The new parent population is then formed from the collection with the most wins from the combined pool. The number of individuals in each tournament is a global parameter of the algorithm ($q = 10$ is a typical setting).

Children are generated asexually, by simply creating a copy of the parent and mutating it. In a similar way to ES, meta EP and Rmeta EP ensure that mutation will favor the search toward the areas of the search space containing better solutions, by the use of variance and covariance strategy parameters held within individuals for each variable. EP allows mutation to modify these parameters, thus permitting self-adaptation to occur. Unlike any of the evolutionary algorithms mentioned so far, EP does not use any form of recombination operator. All search is performed by mutation, following the assumption that mutation is capable of simulating the effects of operators such as crossover on solutions (Fogel 2000).

Child solutions are generated and placed into the population until the population has doubled in size. All new solutions are evaluated, and then the half of the population with the lowest fitnesses are simply deleted. Parents are then picked, offspring generated, and so on. Evolution is typically terminated after a specific number of generations have passed.

All three types of EP can also be *continuous* (Fogel and Fogel 1995), where instead of generating offspring until the population size has doubled and then deleting the unfit half, a single individual is generated, evaluated, and inserted into the population, replacing the least fit individual. This replacement method strongly resembles that used by steady-state GAs (Syswerda 1989).

New advances in EP continue, as this EA is applied to new types of problem. Chellapilla (1997) has recently expanded EP to allow it to evolve program parse trees. Various subtree mutation operators were designed in solving four benchmark problems. He reported that the experiment results showed the technique compares well with EAs that use the crossover operator. The recommended reading section at the end of this chapter suggests books on EP.

Evolutionary Computation Theory

It should come as no surprise that some (but certainly not all) of the theoretical results in natural population genetics also apply to evolutionary algorithms. The field of evolutionary genetics (see Maynard Smith 1989) deals with the spread of favorable genes, selection pressure, migration between populations, and several other issues that concern EA practitioners. There are many differences between the natural evolution and computer evolution studies, however, which means there is little crosstalk between the two fields in terms of theory. The main difference is one of emphasis—a researcher of natural evolution will tend to look for predictions and explanations about the spread of favorable or unfavorable genes given certain underlying mutation rates, for example. An EA developer, however, will be much more interested in how best to set up a range of parameters so as to most effectively attack a specific problem. Partly, this might involve designing specialized operators for the problem at hand, and/or defining specialized genotype-phenotype mappings—but in natural evolution, these aspects are "givens."

However, certain interesting general findings apply, which to some extent have been rediscovered and/or specialized by EA theorists. For example, Price (1970) produced a "covariance and selection" theorem for population genetics, which identifies how the average fitness of the population (or perhaps other measures) will change from one generation to the next, essentially based on the extent to which parents' fitnesses correlate with children's fitnesses.

It is worth taking a look at Price's theorem in a bit more detail. The key aspect of the theorem is that it explicitly identifies the covariance of parent and child fitnesses as an important factor. Covariance is a simply defined mathematical concept. Assume for simplicity that we are only dealing with mutation operators, and that we denote chromosomes by x_1, x_2, x_3, . . . , and their fitnesses by $f(x_1)$, $f(x_2)$, $f(x_3)$, and so forth. Assume further that we have a set P of parent chromosomes, and a set C of child chromosomes, in which each child c_i L C is a mutant of parent x_i L P. The covariance (also called the correlation coefficient) of the parent and child fitnesses is

$$\frac{1}{|P|}{}_{i}\;(F(x_i)-x_m)(f(c_i)-c_m)$$

where x_m is the mean fitness of the parents, and c_m is the mean fitness of the children. If there is absolutely no correlation between a parent's fitness and the fitnesses of its potential children (i.e., inheritance means nothing), then the covariance will be around zero. If parents tend to have children with the same fitnesses as themselves, then covariance will be close to 1. If children tend to "rebel"—fit parents have unfit children, and vice versa—then covariance will be negative, and reach –1 at the extreme of rebelliousness.

It is easy to see how crucial this measure is in any evolutionary system. For example, imagine we are trying to evolve a nice melody, using a collaborative system in which a human listens to each of a population of computer-generated melodies and identifies just a few of them as good parents that the computer will then use to generate the next batch. If there is strong negative covariance between parent and child fitnesses, then this method of selection will be counterproductive—all that the user has done is pick out precisely those chromosomes whose children will be pretty poor! In fact, assuming that the system does nothing sensible to avoid it—such as making sure the best few are always retained in the population—the population of melodies will simply get worse and worse. If instead there was no correlation at all between parent and child fitnesses, we can expect selection to have no effect at all. There is no difference in the kinds of fitnesses we can expect from the children of fit versus unfit parents. The more usual situation, of course, is that there is a positive covariance. That is, fit parents tend to have fit children, and poor parents tend to have poor children. So, to have a good chance of finding things that improve on the best fitness so far, it makes sense to focus on fitter parents—in other words, it makes sense to use a selection and reproduction strategy that implements the "survival of the fittest" idea.

Price's theorem is special in that it makes all this mathematically explicit and clear, and (with only minor reservations) applies equally well to natural and computer evolution. With lots of notation cut away, we can simplify the theorem itself to the following statement:

$$F_{t+1} = R_{t+1} + C_t$$

in which F_{t+1} denotes the mean fitness of the population in the next generation, R_{t+1} is the mean fitness we would expect in the next generation if there was *no* selection pressure (i.e., if parents were selected entirely at random), and C_t is the covariance measure we have already discussed. So, imagine we have a positive correlation between parent and child fitnesses; R_{t+1} can certainly be expected to be no smaller than F_t, and so the theorem clearly indicates that fitness will improve from one generation to the next. Better still, Price's theorem need not refer to fitness per se. F_t could be any measure of the population we like, such as the proportion of chromosomes with a certain gene.

Although Price's theorem is very general and at least assures us that our populations will tend to improve over time, EA researchers and practitioners tend to demand more from the theoreticians in the field; in particular, it would be nice to know if we could be sure that an EA would always produce a result within a certain amount of the best possible solution and how to set up the parameters of an EA to ensure this. Well, there is no proof that even the simplest GA will always converge to an acceptable solution to any given problem in finite time. The most well known attempts so far to theoretically understand evolutionary algorithms, and to build foundations upon which such questions may be addressed, are based on Holland's schema theorem (Holland 1975) and the associated building block hypothesis (Goldberg 1989). A variety of alternatives exist (e.g., Kargupta 1993; Harris 1994), but we will say a little here about the schema theorem, since it is a widely used and accessible route toward understanding the behavior of certain EAs.

A *schema* is a "similarity template" that describes a set of chromosomes. For example, if our chromosomes are binary strings of length five, then the schema 1*011 describes (or "matches") the set of two strings {10011, 11011}—hence, the "*" is interpreted as a *don't care* symbol. The schema *101* describes four strings {01010, 11010, 01011, 11011}. In general, for alphabet of cardinality (number of characters allowed per gene position) k and string length l, there are $(k + 1)^l$ possible schemata (Goldberg 1989).

The *order* of a schema is the number of fixed characters in the template; for example, the order of schema *1*110 is 4, and the order of schema *****0 is 1. The *defining length* of a schema is the distance between the first and last fixed character in the template; for example, the defining length of 1****0 is 5, the defining length of 1*1*0* is 4, and the defining length of 0***** is 0. Finally, we can associate a *fitness* with any schema. The fitness of a schema is simply the average fitness of the chromosomes in the set it describes. For example, the fitness of 10111 is simply the fitness of the chromosome 10111 itself. But the fitness of 1*101 is the mean of the fitnesses of the two chromosomes 10101 and 11101.

Holland's schema theorem states that the action of reproduction (copying, crossover, and mutation) within a genetic algorithm ensures that schemata of short defining length, low order, and high (relative) fitness increase within a population (Holland 1975). Such schemata are known as *building blocks,* and the building block hypothesis (Goldberg 1989) suggests that genetic algorithms are able to evolve good solutions by processing building blocks, and in particular by making use of crossover to combine different building blocks to occasionally produce considerable improvements in fitness. It turns out (see Altenberg 1995) that this is a special case of Price's covariance and selection theorem. As long as building block fitness is positively correlated with chromosome fitness, we can be sure that good building blocks will increase in number from one generation to the next.

There is one major difficulty with this theorem, however, and naturally with Price's theorem too. All that can be said is what will happen from one generation to the next—that is, what the generation at step $t + 1$ will be like given lots of information about the generation at step t. The problem is that the terms on the right in Price's theorem (see above) are far from constant. The covariance between parent and child fitnesses, for example, may be entirely different at generation $t + 1$. It is easy to see this if we consider what Price's theorem has to say concerning a population in which all chromosomes already have the best possible fitness. Obviously, fitness cannot improve in future generations in this circumstance, and this is clearly because there is now a negative covariance, caused by the fact that no child will be fitter than its parent, but many will probably be less fit. The awkward fact is that we can expect the covariance term and the "random" term to both vary at every generation; whether we are talking about fitness or building blocks, this fact rather limits the general applicability of Price's theorem and the schema theorem.

Nevertheless, sophisticated theoretical research to investigate the behavior of the various EAs, for different problems is growing rapidly and making progress. For example, careful analyses of the transmission of schemata have been made (De Jong 1975; Kargupta 1993), and the use of Walsh function analysis (Deb, Horn, and Goldberg 1993) and Markov chain analysis (Horn 1993) has led to the identification of some "deceptive" and "hard" problems for GAs (Deb and Goldberg 1993). In Chapter 4 of *Evolutionary Design by Computers* (Bentley 1999a), David Goldberg summarizes some of the significant advances in the understanding of GAs made to date.

Because there are significant differences between the representation and operators of GP and other EAs, it is unclear whether schema theorem type analyses can be applied to GP. Koza (1992) attempts to achieve this by defining a schema to be the set of all individual trees from the population that contain, as subtrees, one or more specified subtrees. So for GP, disruption is smallest and the deviation from the optimum number of trials is smallest when the schema is defined in terms of a single compact subtree (Koza 1992). As long as crossover is not too disruptive, the fittest of such compact subtrees should grow exponentially within the population and be used as building blocks for constructing fit new individuals. Unfortunately, it seems that crossover in GP is often too disruptive for such theories to be applicable. Definitions of "schema" and the schema theorem are still the subject of much research in the GP community (O'Reilly and Oppacher 1995; Poli and Langdon 1997b).

Theoretical studies centered on ES and EP have tended to be quite independent of schemata-style analysis, instead concentrating on how to set up parameters such as mutation rate and how best to organize self-adaptation. The theoretical roots of ES were developed in the 1970s and apply largely to the simplest form of the algorithm, in this case the (1+1)-ES with no self-adaptation. Rechenberg (1973) analyzed the convergence rates of ES for two test functions and calculated the optimal standard deviation and probability values for a successful mutation regime. From this he formulated the *1/5 success rule:*

The ratio of successful mutations to all mutations should be 1/5. If it is greater than 1/5, increase the standard deviation; if it is smaller, decrease the standard deviation.

Born (1978) subsequently formally proved that this simple form of (1+1)-ES will result in global convergence as long as there is a positive standard deviation. Other theoretical analyses have shown that the introduction of populations in ES causes a logarithmic speedup in evolution over the simple (1+1)-ES (on strongly convex functions).

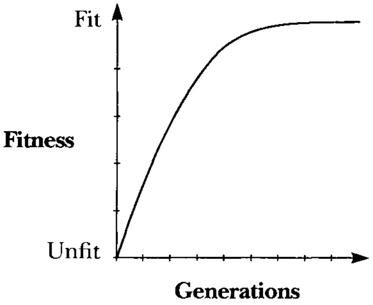
As Bäck (1996) describes, Schwefel derived an approximation theory for the convergence rate of the simplified (μ, λ)-ES and $(\mu + \lambda)$-ES (one standard deviation for all object variables; no crossover nor self-adaptation). More recently, Rudolph (1996, 1997, 1998) has used Markov chains to investigate the convergence theory in EAs.

Although ES theory can be applied with little modification to EP (Bäck 1996), Fogel has performed independent analyses of various forms of EP, finding that mutations should increment or decrement the value of a decision variable by no more than the square root of the fitness score of the solution. Fogel also proved that simple EP will converge with a probability of one; however, convergence rates (time taken to converge) and other significant features of EP remain unsolved (interested readers should consult Fogel 1992b). Recent theoretical and empirical work by Yao and Liu (1996) has identified an alternative type of mutation operator to use within EP—this uses a Cauchy instead of Gaussian (normal) distribution of deviations, essentially meaning that although there remains much higher chance of small deviations, large deviations will be much more common than with the Gaussian operator. In Yao, Lin, and Liu (1997), an analysis based on the study of neighborhood and step size of the search space is performed to compare these two mutation operators. Their work provides a theoretical explanation, supported with empirical evidence, of when and why Cauchy mutation is better than Gaussian mutation in EP search. The long jumps provided by Cauchy mutation increase the probability of finding a near-optimum when the distance between the current search point and the optimum is large. The Gaussian mutation, on the other hand, is a better search strategy when the distance is small.

A study by Gehlhaar and Fogel (1996) also indicates that the order of the modifications of object variables and strategy parameters has a potentially strong impact on the effectiveness of self-adaptation. It is important to mutate the strategy parameters first and then use them to modify the object variables. In this way, the potential of generating good object vectors that are associated with poor strategy parameters can be reduced. Thus the likelihood of stagnation can be reduced.

Experimental comparisons between the different types of EP are ongoing. For example, EP with self-adaptation has now been reapplied to the original task of evolving FSMs (Fogel, Angeline, and Fogel 1995, Angeline and Kinnear 1996). Each state in an FSM was associated with mutation parameters to guide which component of the FSM was to be mutated and how to perform the mutation. Angeline and Kinnear (1996) reported that EP with self-adaptation performs better than a standard EP when solving a predication problem.

Finally, while theoretical studies are necessarily attuned to simple EA designs and usually unrealistic test problems, extensive empirical studies have revealed many things that seem to be generally true of all EAs and most real-world applications. For example, evolution tends to make extremely rapid progress at first, as the diverse elements in the initial population are combined and tested. Over time, the population begins to converge, with the separate individuals resembling each other more and more (Davis 1991). Effectively this results in the EA narrowing its search in the solution space and reducing the size of any changes made by evolution until eventually the population converges to a single solution (Goldberg 1989). When plotting the best fitness value in each new population against the number of generations, a typical curve emerges, such as Figure I.14 (Parmee and Denham 1994).



1.14 Typical curve of evolving fitness values over time.

FIGURE

Generalized Evolutionary Algorithms

The perceptive reader should be able to see that the four major types of evolutionary algorithm we have discussed are really just points in an infinitely large space of possible evolutionary algorithm designs. The division of the evolutionary computation field into these four distinct areas has arisen for historical reasons, but there is certainly no reason to believe that the "best" EA for a particular problem is closer to one of these four points than to another. These days, a modern breed of evolutionary computation researchers is assiduously exploring the space of EA designs and thereby producing algorithms that do not neatly fall into any of the four main groups. This may be because aspects of two or more types of EA are combined—for example, using an EP-style algorithm to evolve tree-structured programs, such as Chellapilla (1997)—or because the algorithm is simply radically different from any of the major four, such as the population based incremental learning algorithm (Baluja and Caruna 1995). Bentley (1999c) attempts to outline the dimensions of this space of possible EA designs in the context of a general architecture for evolutionary algorithms (GAEA). Here, we rapidly summarize the shape of the GAEA and extend it a little.

We start by noting what all points in EA design space should have in common. As stated by the theory of Universal Darwinism (Dawkins 1983), necessary criteria for evolution to happen are the presence of *reproduction, inheritance, variation,* and *selection.* That is, any point in EA design space must represent an algorithm in which new individuals can be generated (reproduction), these new individuals are often largely similar to old ones (inheritance), but with some differences (variation), and there is some bias toward inheritance from "better" individuals (selection).

Selection, in particular, is both critical to the success of an EA (i.e., without selection, there is no force to drive the population toward improving fitness) and can be implemented in such an algorithm in several ways. Consider the real-world process of selective cattle breeding, for example. We can implement selection either by making sure we only allow better cows to breed or by letting all cows breed, but ruthlessly culling low-milk yield calves, or both. (If we could change the number of calves produced by a cow each time, we would also have a third way to induce selection pressure.) Something akin to the former technique is what "selection" tends to denote in the EA world, but EAs may use all of these techniques.

Points in EA design space will also all share certain other features. Namely, there will be some way of initializing the population, some way of evaluating the fitness of candidate solutions, and some way of determining when to stop. In other words, dimensions in the space of EA designs will include *initialization, evaluation,* and *termination.*

There are two further, less obvious, dimensions in which points in EA design space may vary that are part of Bentley's (1999c) GAEA: *mapping* and *moving.* Mapping refers to the distinction between phenotype and genotype. At first sight it may seem sensible to simply package this up with the term *evaluation,* since in all cases we expect the evaluation function to somehow "decode" the genotype (thus finding the phenotype it maps onto) and then calculate the fitness of that phenotype. However, emerging areas in the EA world such as embryology are looking into very sophisticated genotype/phenotype mappings, allowing highly complex solutions to be specified using a compact set of instructions. Certainly, the mapping stage may vary independently of the fitness calculation, hence its presence as a standalone feature of the GAEA.

The moving dimension refers to the organization of population dynamics. It has been found highly successful to structure a population of chromosomes into independently evolving subpopulations (Whitley and Starkweather 1990) or localized, overlapping subpopulations (see Collins and Jefferson 1991; Davidor 1991), in which separate parts of the population evolve (semi-)independently, but occasional migration moves chromosomes from one part of the population into another. This allows separately evolving individuals to occasionally share useful genetic information, reducing premature convergence within species, reducing the number of evaluations needed, and improving the quality of solutions evolved.

There are two further important dimensions in which evolutionary algorithms can vary, and which we herein add to the GAEA architecture of Bentley (1999c). These can be called *operator adaptation* and *restart.* A typical *applied* EA—that is, one that has been carefully designed and tailored toward addressing a particular application—might use a large collection of specialized crossover and mutation operators. A problem arises (in all EAs, but much more so here) regarding what are called the "operator rates"—that is, how often to apply each individual operator. A suitable answer to this problem is to let the algorithm work it out; that is, the rates of application of an operator will evolve and adapt over time, changing according to their current level of success or failure. There are many possible schemes for this adaptation process (see Tuson and Ross 1998). Finally, *restart* refers to a scheme employed by an algorithm to effectively start again when the population seems not to be getting anywhere, but not quite from scratch, before its time is up. The well-known GENITOR II algorithm (Whitley and Starkweather 1990) employed a specific restart mechanism, whereas in the CHC algorithm (Eshelman 1991, and also see Chapter 13), the restart method is central to the algorithm, being the only place in which mutation is ever applied.

In summary, a generalized evolutionary algorithm will have something to say (although in some cases there may be nil returns) about each of the following components: *initialization, mapping, evaluation, selection, operator adaptation, fertility, reproduction, restart,* and *termination.* Most of these stages are described much more fully in Bentley (1999c). Here, our goal is simply to give some idea of the massive degree of unexplored territory in the space of possible evolutionary algorithm designs. It is unknown whether or not there are particularly fruitful but as yet untouched regions of this space, but its size seems to suggest (unless significant advance in theory can obviate it) that research in this area will take quite a while to fizzle out, if ever!

From Evolutionary Algorithms to Creative Evolutionary Systems

This section of the chapter has introduced the concept of using computers to search for good solutions in a search space. The parallel searching mechanism used by evolutionary algorithms was described, and the four major EAs were summarized: genetic algorithms, genetic programming, evolution strategies, and evolutionary programming. The section concluded by briefly describing a general architecture for evolutionary algorithms, emphasising that all EAs are fundamentally points in a very large space of potential EA designs.

Having now explained how computers are used to perform evolution, the following section describes how we can make computers creative by using evolution and how evolution can help us to be creative.

CREATIVE EVOLUTIONARY SYSTEMS

Using evolution as a simple function optimizer is one thing, but how do we enable evolution to handle creativity? What is a creative evolutionary system, anyway?

Put simply, a *creative evolutionary system* is a computer system that makes use of some aspect of evolutionary computation and is designed to

- aid our own creative processes, and/or
- generate results to problems that traditionally required creative people to find solutions.

In achieving these goals, a creative evolutionary system may also appear to act "creatively." For example, a system might find highly innovative and novel solutions, or it might combine two very different ideas to make something new. The debate on whether evolutionary computation is really creative or not will be presented in the last section of this chapter, but first we must explore creative evolutionary systems.

Again it is worth stressing that such systems are not intended to replace people, but increase productivity and creativity by allowing people to explore more and a wider variety of solutions than they could without such computer systems. We will first look in more detail at how and why these techniques were developed.

Background

Many of the advances in creative evolution grew up in the field of evolutionary design (Bentley 1999a). It is easy to see why this has happened. Design problems such as architecture, engineering design, and aesthetic design are horribly complex, with huge numbers of (often conflicting) objectives, many constraints, and often thousands of parameters (Parmee 1999). But the most difficult aspects of these design problems are *people*. Designs are usually used by people—we live in architecture, we use and interact with the things around us. We like and dislike things almost at random, and as fashions change, so our preferences and requirements change. This means that a design specification will usually be a moving target (French 1999). It will have many unknowns, and the few things that are true one week will not necessarily be true the next week.

Designers take such things for granted. They know that their designs will be revised and modified many times until clients are satisfied (or until time or costs prevent further changes). And architecture is perhaps the most extreme type of design in this respect. There are so many different rules, opinions, preferences, and materials that, for every new building, there will be an infinite number of possible design solutions. Exploring these solutions forms part of the difficult job of being an architect. Consequently, it is no coincidence that the first forays into explorative and generative evolutionary design were made by architects. Some of the earliest work was performed by John Frazer, who spent many years developing evolutionary architecture systems with his students (Frazer 1995). He showed how evolution could generate many surprising and inspirational architectural forms and how novel and useful structures could be evolved. His methods often involved the use of bricklike components such as cellular automata, which were evolved and sometimes wrapped in surfaces to generate smooth exteriors. His chapter, "Creative Design and the Generative Evolutionary Paradigm" (Chapter 9), describes some of these in more detail. In Australia, the work of John Gero and colleagues such as Michael Rosenman (both architects) also investigated the use of evolution to generate new architectural forms. This work concentrates on the generation of new floor plans for buildings, showing over many years of research how explorative evolution can create novel floor plans that satisfy many fuzzy constraints and objectives (Gero and Kazakov 1996). They even show how evolution can learn to create buildings in the style of well-known architects (Schnier and Gero 1996).

Designers and architects still remain at the forefront of this area of research. Today increasing numbers are creating evolutionary architecture systems capable of generating novel forms, structures, buildings, and even towns. For example, Paul Coates of the University of East London has shown how evolution can generate coherent plans for housing estates and buildings, as well as innovative building exteriors (Coates 1997). Celestino Soddu of Italy uses evolution to generate everything from novel table lamps to castles to three-dimensional Picasso sculptures (Soddu 1995, and Chapter 2). The work of Ian Parmee at the University of Plymouth has revealed a variety of methods for handling the complex and fluid nature of engineering design problems over the years (Parmee 1999).

Artists are also keen researchers into creative evolutionary systems. Dawkins's "biomorphs" inspired the well-known work of William Latham and

Stephen Todd, who had considerable success in evolving artistic images and animations (Todd and Latham 1999). The work of Karl Sims was also inspired by Dawkins and also showed the astonishing complex and aesthetically pleasing images that evolution could generate (Sims 1991). Evolutionary art systems have now become very popular, with many programs available and some now being incorporated into art and CAD packages (Rowbottom 1999). For more details, see Part 4 of this book. In addition, Mitchell Whitelaw explores more of these in his chapter, "Breeding Aesthetic Objects: Art and Artificial Evolution" (Chapter 3).

It is interesting to note that all of these examples of evolutionary systems used evolution more as an explorer than an optimizer. Normally guided by a human, the software is used to investigate many possible solutions, to provide inspiration, and to give a feel for the range of useful solutions.

Although producing impressive results, such software always received some criticism that the presence of a human to guide the direction of search by evolution was the key to the success of these systems. Peter Bentley's work in this area, and the work of others such as Adrian Thompson and John Koza, provided some of the first convincing demonstrations that evolution was capable of innovation without human guidance (the second type of creative evolutionary system). Bentley showed how evolution would find surprising and creative solutions to design problems, even when only software guidance was provided. By telling the computer the desired function in the form of a set of evaluation routines, but not anything about the design itself, the human was removed from the loop and the power of explorative evolution was shown conclusively (Bentley and Wakefield, 1995, 1997a, 1997b; Bentley 1996). Thompson also demonstrated the power of evolution, this time to generate novel electronic circuit designs that were tested using field-programmable gate arrays (Thompson 1995). His chapter, "The Sound Gallery—An Interactive A-Life Artwork" (Chapter 8), explains how these techniques can be used to generate music. In a similar vein, John Koza has been demonstrating the use of genetic programming to find novel computer programs for some years—to use computers to program themselves (Koza 1992). Work such as this began the recent change in thinking about evolution. No longer were we content to regard an evolutionary algorithm as "another optimizer." Our evolutionary programs were now independently solving problems for us and finding creative solutions that surprised us.

Since then, research in this area has expanded rapidly. Adrian Thompson spends his time trying to develop ways to analyze how his evolved circuits work (Thompson and Layzell 1999). Julian Miller has expanded upon Thompson's work, showing how evolution can create circuits that seem to work on entirely new principles; see his chapter, "The Genetic Algorithm as a Discovery Engine: Strange Circuits and New Principles" (Chapter 18). John Koza has announced the completion of a 1,000-processor Beowulf supercomputer, which will be used for the sole purpose of "using genetic programming as an automated 'invention machine' for creating new and useful patentable inventions" (Koza 1999). His group has recently demonstrated the use of evolution to generate many different types of analog circuit, many of which mirror or outperform our best human-created circuit designs (Koza et al. 1999). Chapter 10, "Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results," provides more information on this. Jordan Pollack has shown how evolution can generate highly novel structures such as bridges and cranes (Funes and Pollack 1999) and has now begun to use these methods for the evolution of robot bodies, as described in his chapter, "Evolutionary Techniques in Physical Robotics" (Chapter 21). Karl Sims used evolution to create the bodies and brains of "virtual creatures" capable of swimming, walking, jumping, and competing in virtual environments (Sims 1999). Husbands et al. (1996) used evolution to generate novel propeller-like forms. Many research departments around the world use evolution to generate new and highly complex neural networks for various applications. Bentley's work in this area continues, as he uses evolution to compose music that cannot be distinguished from human compositions (Bentley 1999b).

Now that we have a feel for the whys and hows of creative evolutionary systems, let's look a little more closely at the two types. We have seen that these systems can aid our own creative processes and can generate results to problems that traditionally required creative people to find solutions. Let's first examine more closely how evolution can be used to aid our creativity.

Aiding Our Creativity

Richard Dawkins was one of the first to show the world how evolution on a computer could be combined with the aesthetic preferences of a user to generate an almost infinite array of pleasing or interesting forms (Dawkins 1986). But this form of creative evolutionary system existed long before computers were ever invented. Horses, cows, sheep, dogs, cats, guinea pigs, birds, wheat, corn, roses, trees—all belong to the enormous list of "creatively modified" forms of life. Through selective breeding, where a person selects the parents of each generation based on his/her aesthetic judgment or functional requirements, we have enabled evolution to generate a wide range of imaginative, beautiful (and sometimes ugly) forms. In both nature and computer science, evolution provides us with the novelty, showing us new and subtly different solutions every generation. We act as the judge for this kind of creative evolutionary system, choosing some solutions with certain features, rejecting others. As we watch, we see innovations and mistakes unfold before us, and our imagination, our creativity is engaged. The partnership of evolution and human judgment enables a richer and more diverse kind of creativity.

This, then, is the essence of this type of creative evolutionary system. The user of the system is the judge, who chooses the short-term path of evolution. With real persistence, the user may be able to make the right choices to reach a specific final goal, but usually in most of these systems users quickly become very reactive decision-makers, basing their choices on what they see immediately in front of them and not on what they hope they might end up with. This is only natural, for it is evolution that provides the choices, and although it *might* eventually provide every solution possible, it rarely gives the user an expected result. And it is this very feature that makes creative evolutionary systems so useful. If users already know what they would like to see, it is rare for an evolutionary system to be needed in order to obtain it (Peter Hancock provides the exception to this in Chapter 16, "Evolutionary Generation of Faces"). Normally it is the very unpredictability of evolution that helps stimulate the creativity of people. By constantly throwing up unexpected and unlikely solutions, users are forced to react, change their minds, explore new possibilities. This type of creative evolutionary system can enhance the creativity of people.

Collaborative Evolutionary Algorithms

So we make this kind of creative evolutionary system by adding human interaction to the evolutionary process. Our judgment must contribute to or replace the fitness functions (or any other part of the evolutionary algorithm that helps generate selection pressure). When we modify EAs in this way, we call them interactive evolutionary algorithms, or *collaborative* evolutionary algorithms.

There are a large number of different approaches used in this area. For example, Furuta, Maeda, and Watanabe (1995) allowed users to judge the aesthetics of evolving bridges in addition to their structural evaluation and employed "psychovectors" to quantify aesthetic factors. Others use multiobjective optimization methods to combine user input and fitness functions (Bentley 1999a), and some use fuzzy logic to aid the input of knowledge into the search (Parmee 1999). Most evolutionary art systems will present users with some or all of the evolving population and allow them to rank or vote on the quality of the images (Rowbottom 1999). Some, like the system described in Chapter 8, even allow users to "vote with their feet" and physically move themselves toward evolving solutions that they prefer.

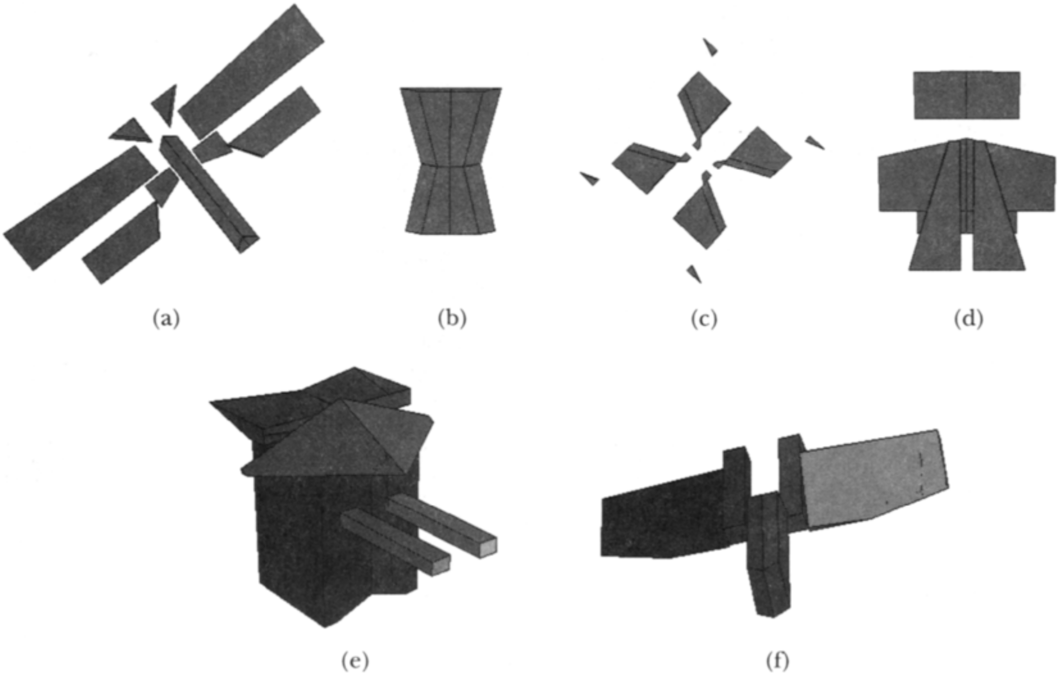
By adding human guidance into evolutionary search, we gain certain advantages, but also incur certain disadvantages. The advantages include the following:

- Good searching ability—if evolution seems to "get stuck" (converge) on a certain type of solution, the users can alter their guidance and force evolution to try alternatives.
- A wide range of different solutions—the longer users "play" with such systems, the more solutions they will see.
- The ability to evolve solutions for which there is no clear fitness function—if you cannot work out why something looks nice, sounds good, or is fashionable, you cannot use software evaluation to automate the fitness evaluation. Also, if the objectives are highly variable as well as being subjective, human interaction permits quick re-evolution without the need to rewrite fitness functions.

The disadvantages of collaborative systems include the following:

- Speed—humans can only judge solutions at a very slow rate, so evolution may take rather longer.
- Consistency—humans often change their minds, get distracted or bored, or become influenced by other solutions, so they will not judge solutions consistently.
- Coverage—in order to provide effective guidance, users need to judge most (or ideally all) different solutions in an evolving population. But for hard problems, large population sizes and many generations may be required. Humans simply cannot cope with the thousands of evaluations that are necessary for a typical evolutionary run.

Of course, it is also very important to have a good representation that enables evolution to find and define a huge range of alternative solutions; we will look at issues of representation later. But the addition of user input into an evolutionary algorithm is so easy, and the results often so good, that it is surprising how few researchers actually permit it. To illustrate how simple it can be, Bentley's *genetic algorithm designer*, GADES (Bentley 1999a), was modified for this introduction. A new evaluation module was added that simply presented each member of the evolving population to the user in turn and asked for a fitness score to be input (a number between 0 and 9). With a population size of 10 individuals, slightly higher mutation rates than normal (to slow convergence), and



I.15 FIGURE "Aesthetically pleasing" (or at least interesting) shapes evolved by the interactive GADES: (a) dragonfly, (b) drum, (c) flower, (d) robot, (e) 3D evil robot, and (f) 3D airplane.

about 30 or 40 generations, a wide variety of "interesting solutions" were evolved. Because of the time needed to judge the solutions, each took around half an hour. Figure I.15 shows how creative the results of evolution were, when combined with human guidance.

Generating Creative Results

The second way in which these systems can aid creativity is by taking over more of the role of the creative person and actually providing solutions to a problem automatically. It is hard to automate the judgment of aesthetic preferences (although it can be done in a limited way), but we have had considerable success in evolving solutions that fulfil a desired function. We can now evolve aerodynamic cars, highly novel circuit designs, architecture, and music without the need for constant human guidance. This allows more solutions to a problem to be found more quickly and also provides inspiration and information about what kinds of solutions exist for the user. Problems of "design fixation" are removed, and radical new and more efficient concepts can be found more quickly.

Why is this such a big deal? We have been optimizing solutions for many years, and no one has claimed anything about "creativity" before. The difference is twofold: First, we are now enabling evolution to find solutions to problems that traditionally have been regarded as something of an art form—whether composing music or designing analog circuits. To claim that evolution can now perform as well as the experts for some of these applications is heresy to many people, but as the chapters in this book show, it is beginning to become true (see Chapter 10). The second difference is the way we apply evolution to these problems. Instead of providing an existing solution that is then optimized, most creative evolutionary systems only provide the tools to build new solutions, allowing evolution to discover novelty and innovation by itself. And it turns out that this is rather useful, for the kinds of solution that evolution discovers are often very different than the kinds of things we think of.

Making Evolution More Creative

So how can evolution achieve these marvels? We will discuss exactly what it means to be creative in the last section of this introduction, but one common definition of creativity is *removal of constraints.*

It is no coincidence, then, that when we examine how creative evolutionary systems differ from more traditional evolutionary systems, we see that constraints of one form or another have been relaxed or removed. Some researchers do this quite explicitly: for example, the recent work of Ian Parmee concentrates on the development of interactive evolutionary systems that allow users to relax functional constraints for engineering design problems. As Parmee states: ". . . close individual designer and design team interaction with evolutionary and adaptive computing strategies can result in significant exploration involving off-line processing of initial results and related design information leading to a redefinition of the design environment" (Parmee 1999). In other words, the intention is to allow designers to relax constraints in order to explore more potential solutions. Then with this additional knowledge designers can redefine representations and allow evolution to provide newer or different solutions more appropriate for the application. But there are other, more important constraints that must be considered if we are to enable creative evolutionary systems to generate novelty.

Traditional implementations of evolutionary search suffer from the same fundamental drawbacks as all conventional search algorithms. They rely on a good parameterization to permit them to find a good solution. If we are optimizing a propeller blade, but the parameterization does not permit the width of the blade to vary, then the computer will never be able to find solutions with different widths. If there is no parameter for something, then the computer cannot modify it. Evolution, like all search algorithms, is limited and constrained by the representation it can modify. So it is clear that while relaxing functional and parameter constraints will permit evolution to arrive at a larger diversity of solutions, only by modification of the representation can evolution go beyond optimization. It seems that the remarkable results obtained by this type of creative evolutionary system require the removal of constraints within the representations, and not only the fitness functions. And by using a different type of representation, we can cause this to happen automatically.

Component-Based Representations

Traditional views of an evolutionary algorithm regard this search technique as an optimizer. A better term is actually "satisficer" (Harvey 1997). Evolution never tries to find globally optimal solutions. It merely propagates improvements through the population. In doing so, evolution walks a mysterious and winding path through the search space. Sometimes this path may reach a dead end (premature convergence). Sometimes the path may go around in circles. And occasionally the path may lead to a global optimal—but there is no guarantee of this. The wanderings of evolution are like those of an explorer.

But exactly what does evolution explore? This is determined by the representations it uses. With a fixed-length parameterization, explorations do resemble optimization. For example, if there are only three parameters defining the solution and a single fitness function, evolution will naturally behave in much the same way as an optimizer—it will find the best values for those three parameters such that the solutions are as fit as possible. But with a different kind of representation, the behavior of evolution changes. When the parameters do not define the solution directly, when they define a set of components from which the solution is constructed, the idea of optimization becomes inappropriate. Now evolution *explores* new ways of constructing the solution by changing the relationships between components. It can vary the dimensionality of the space by adding or removing elements. It can explore alternatives instead of optimizing a single option. When we examine the representations, objectives, and goals of optimization and exploration, the difference between these approaches becomes clearer.

Optimization

A knowledge-rich encoding of the problem is used; that is, a solution is parameterized using the fewest possible parameters (and thus both minimizing the search space, but also minimizing potential for exploration). Evolution is used only to find the best parameter values within that parameterization. Objective functions are normally predefined and fixed. The goal is to find a global optimum or near global optimum with the least amount of computation.

Exploration

A knowledge-lean representation is used, and instead of a parameterization of a solution, a set of low-level components is defined. Solutions are then constructed using the components, allowing exploration (sometimes at the expense of size of search space and ability to locate optima). Objective functions may vary at any time. The goal is to identify new and interesting solutions—normally more than one is desirable—and these solutions must be good. However, finding global optima may be undesirable, impractical, or even impossible.

This difference between optimization and exploration becomes very obvious when the literature in this area is examined. It becomes evident that every creative evolutionary system uses this idea of component-based representation in some way. Researchers who evolve architecture use evolution to manipulate components such as walls or bricks. Researchers who evolve electronic circuits use evolution to modify components such as logic gates, transistors, and capacitors. Artists use evolution to create art out of primitive shapes, such as swirls, spheres, and tori, or out of mathematical functions such as cosine, arctangent, and factorial. Likewise, computer scientists using GP evolve programs from components contained within the function and terminal sets. Bentley's work uses cuboid blocks to construct designs, fuzzy logic functions to construct fuzzy rules, and musical notes to construct compositions. Every explorative evolutionary system relies on some kind of component-based representation. Figure I.16 illustrates this idea, showing how components allow increased freedom for evolution. The shape in Figure I.16(a) uses minimal parameters and is knowledge-rich (the height of the shape and the fact that the 2D outline is swept to give a 3D shape are assumed by this representation). The shape in Figure I.16(b) is constructed by wrapping an envelope around a collection of 3D ellipses. By varying the relative positions and sizes of the ellipses, vastly more innovative and creative forms can be generated.

Evolutionary algorithms, and in particular genetic algorithms and genetic programming, naturally lend themselves to this kind of exploration. It is standard practice to use evolution to manipulate coded versions of solutions (genotypes) and to map these onto actual solutions (phenotypes). This use of genotypes and phenotypes means that the distinction between, and use of, components of solutions and complete solutions is natural to this field.

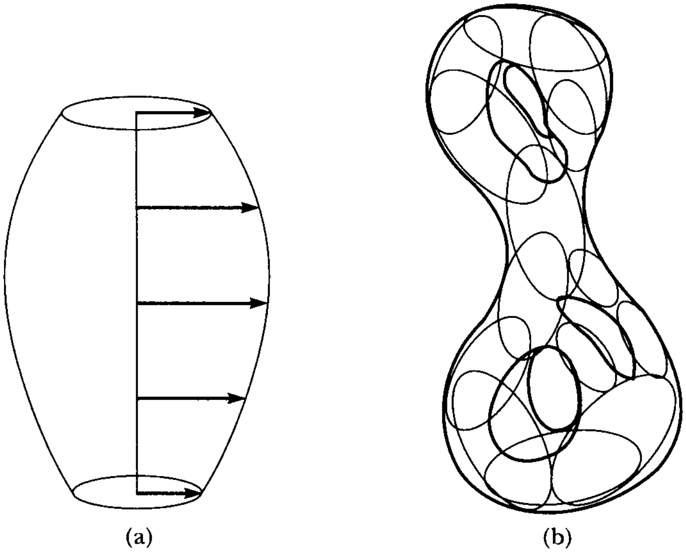


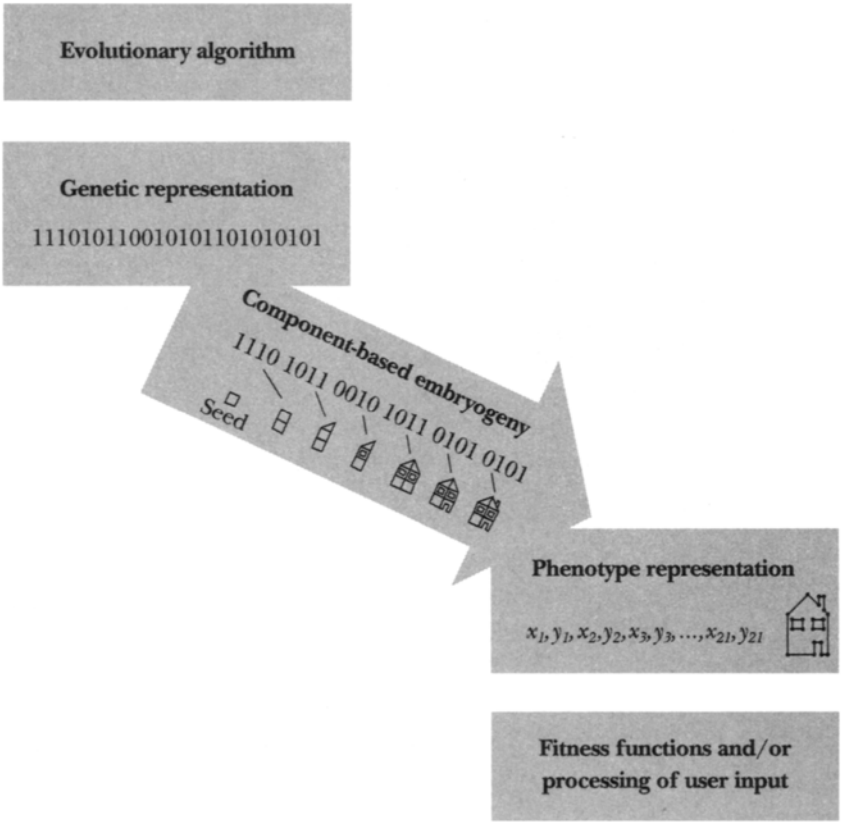
FIGURE I.16 (a) Optimization of fixed parameterization versus (b) component-based exploration.

A Framework for Creative Evolution

From this investigation of creative evolutionary systems, we can construct a framework that contains five elements (see Figure I.17):

- An evolutionary algorithm
- A genetic representation
- An embryogeny using components
- A phenotype representation
- Fitness function(s) and/or processing of user input

To summarize, a creative evolutionary system requires some kind of evolutionary algorithm to generate new solutions. The algorithm modifies genotypes defined by the genetic representation, which should be designed to minimize disruption caused by the genetic operators. An embryogeny (or mapping process) should decode the genotype and, using some kind of components, should construct the phenotype. The phenotype representation should be designed such that it permits quick and efficient evaluation by the fitness function(s). It is


Evolutionary algorithm
Genetic representation
1110101100101011010101010
Component-based embryogeny
1110 1011 0010 1011 0101 0101
Seed
Phenotype representation
$x_1, y_1, x_2, y_2, x_3, y_3, \ldots, x_{21}, y_{21}$
Fitness functions and/or processing of user input

1.17 FIGURE The five elements of the framework for creative evolutionary systems.

likely that the evolutionary algorithm, the genetic representation, and, to some extent, the embryogeny will be generic and suitable for reuse for most problems without modification. The phenotype representation and fitness functions will usually be specific to the current application of the system. The fitness functions may also be designed to process human input rather than actually evaluate the solutions directly. The following sections explore the elements of the framework for explorative evolutionary systems in more detail.

Evolutionary algorithm

The evolutionary algorithm forms the core of any evolutionary system. As described earlier, there are four main EAs in use today: the genetic algorithm, genetic programming, evolutionary strategies, and evolutionary programming.

Sadly, only the GA and GP are commonly used for the exploration of creative solutions. The reason for this can perhaps be found in the way these algorithms work. The genetic algorithm maintains genotypes and phenotypes, with a mapping between the two. As described earlier, this distinction has helped to encourage some GA researchers to use component-based genotype representations that map onto the phenotype representations, thus allowing explorative evolution to begin. In the same way, genetic programming also makes use of genotypes (this time with tree structures) that are mapped onto phenotypes such as programs, images, or circuits. GP has the advantage that its genetic representation *requires* the use of smaller components (in the function and terminal sets), so all applications of GP demonstrate the explorative power of evolution. This may explain why the first notion of "invention machine" came from John Koza, the inventor of GP—his algorithm ensures that explorative evolution will always take place. In contrast, algorithms such as evolutionary strategies and evolutionary programming make no distinction between genotype and phenotype. By directly modifying the solution and with no provision for mapping to new representations, these approaches make the use of components to construct solutions difficult to implement and hence almost unheard of in this field. Nevertheless, there is nothing inherent in any evolutionary algorithm that prevents its use in a creative evolutionary system.

Within any evolutionary algorithm there are other issues that must be tackled. Handling multiple objectives, multimodality, noise, premature convergence, and fuzzy or changing fitness functions must all be considered. Solutions to all of these problems, using ideas such as Pareto optimality, region identification, speciation, variable or directed mutation rates, and steady-state GAs are now emerging in evolutionary computation (Bentley 1999c; Parmee 1999; Vavak and Fogarty 1996). These issues, although important, are not the most significant consideration for creative evolution. Indeed, even the choice of evolutionary algorithm (or indeed any other search algorithm) is secondary to the representations, for it is the representations that permit evolution to explore.

Genotype Representation

The genotype representation defines the search space of the algorithm. A poor representation may enumerate the space such that very dissimilar solutions are close to each other, making search for better solutions harder. For creative evolutionary computation, where genes will represent (directly or indirectly) a variable number of components, the search space is typically of variable dimensionality, thus making its design even harder (Gero and Kazakov 1996).

There are also other problems. Because of the use of components to represent solutions, the likelihood of epistasis dramatically increases. Not all component-based representations will have this effect (e.g., a one-to-one mapping with a voxel representation allows both exploration and zero epistasis). However, most components are inherently linked to their companions for the solution to work as a whole. A circuit relies on the links between its components, a melody relies on links between notes, a house relies on links between walls, a program relies on links between commands. These linkages all mean that corresponding genes become epistatically linked, resulting in potentially serious problems for evolution. With polygeny so prevalent in these problems, great care must be taken in the design of the genotype representation and corresponding genetic operators to minimize the disruption of inheritance.

Practitioners of GP have long been aware of these problems, and many solutions are now in existence. Modifications can be made to the genetic representation to increase functionality and decrease disruption, for example, encapsulation, ADFs, ADIs, ADLs, and so on (Koza et al. 1999). New genetic operators that enforce typing help ensure that genetic trees are not shuffled too drastically during the production of offspring (Page, Poli, and Langdon 1999).

GAs do not require the use of tree-structured genotypes, so genetic representation-based problems are often less prevalent. GAs can be used to evolve variable-length genotypes and structured genotypes, typically with operators designed to perform crossover only at points of similarity between two parent genotypes (for example, Bentley and Wakefield 1996c; Harvey 1992). Advanced GAs designed to minimize damage caused by disruption of epistatic links between genes have also been demonstrated (Goldberg 1999). In addition, GAs do not suffer from the classic GP problem of bloat, where genotypes tend to increase in size, with redundant genetic material becoming ever greater in solutions. It is clear that the creation of suitable genetic representations and corresponding operators is a considerable problem in its own right. Furthermore, recent research indicates that significant benefits may be gained from using less complex genetic representations and operators, instead making use of embryogenies of greater complexity (Bentley and Kumar 1999).

Embryogeny

An embryogeny is a special kind of mapping process from genotype to phenotype. Within the process, the genotype is now regarded as a set of "growing instructions"—a recipe that defines how the phenotype will develop. Polygeny—phenotypic traits being produced by multiple genes acting in combination—is common. Research in this area has revealed some of the potential of these advanced mapping processes. Advantages include reduction of the search space, better enumeration of the search space, the evolution of more complex solutions, and adaptability (Bentley 1999c).

Embryogenies are widely used for explorative evolutionary systems because they provide the mechanism for constructing whole solutions from components. Three types of embryogeny are used today. The first and most common is *external,* where a programmer writes the software that performs the mapping, and the process cannot be evolved—it is external to the genotype (Bentley 1999c). More recently, *explicit* embryogenies have become popular, with every step of the growth process explicitly held as part of the genotype and evolved (Bentley 1999c). Examples include cellular encoding, used by Koza and team for the evolution of analog circuits (Koza et al. 1999); Lindenmayer systems, used by Coates (1997) for the evolution of architectural forms; and shape grammars, used by Gero for the evolution of floor plans (Schnier and Gero 1996). Despite the considerable success of these embryogenies, they often require complex additions to genetic representations and operators to allow evolution to work. The third type of growth process, the *implicit* embryogeny, has shown the most exciting results and greatest potential in recent work. Instead of evolving the mapping as a set of explicit steps in the genotype, an implicit embryogeny uses a set of rules, typically encoded as binary strings in a GA genotype. For each solution, a "seed" component is created, and then the rules are iteratively applied. Over many iterations, with rules activating and suppressing each other, the growth, position, and type of new components are built up, finally resulting in the development of a complete solution. This emergent growth process shows remarkable properties of scalability, with the genotype describing solutions of increasing complexity without any increase in the number of rules needed; the rule-directed growth process is simply allowed to run longer. This is in contrast to all other approaches, which require significantly larger genotypes to define the increased growth of more complex solutions (Bentley and Kumar 1999).

The process of mapping from genotypes to phenotypes is clearly of importance to the investigation of explorative evolution. Issues of scalibility, evolvability, and biases induced in search have yet to be considered by researchers in any great depth. Increased understanding in this area would benefit both computer science and developmental biology.

Phenotype Representations

Once constructed by the embryogeny, the resulting solution is defined by the phenotype representation. Typically this representation is specific to the current application: if we are evolving circuits, the representation might define networks of connected components; if we are evolving buildings, the representation might define exterior shapes and/or interior walls, floors, and stairs. An important criterion is evaluation. Typically the phenotype representation will be designed to allow direct evaluation by fitness functions, without intermediate transformations or calculations. A poor choice will detrimentally affect processing times and solution accuracies.

The distinction between genotype, embryogeny, and phenotype representations is often blurred in this field. Some GP practitioners regard all three to be the same. Others, such as Jakobi's work on evolving neural networks (Jakobi 1996) or Taura's work on evolutionary configuration design (Taura and Nagasaka 1999), use different and distinct representations for each stage. It is still unclear whether the component growing process of the embryogeny should use the same representation as the phenotype. Should the phenotype be represented by blocks, or should the blocks be merged into a single, whole description of the solution? For example, should an evolved musical composition be represented by a series of notes or by a single, complex waveform in the phenotype? The answer is likely to depend on the fitness functions.

Fitness Functions and/or Processing of User Input

The fitness functions must provide an evaluation score for every solution. For creative evolution computation, this is often a little harder. Typically the use of components rather than a parameterized solution means that early results can be chaotic, to say the least. A design of a car may resemble a shoe; a melody may sound like a burglar alarm. Before evolution has had time to improve these initially random solutions, they can be nothing at all like the desired result. And yet the fitness functions must always be able to provide a fitness score that makes sense. The task is made even harder when unknowns or approximations must be incorporated into the evaluation, or when constraints and objectives are varied to aid exploration. Potential solutions include the use of custom-designed modular functions (Bentley 1996) and fuzzy logic (Parmee 1999) to cope with such problems.

As discussed earlier, many creative systems use human input to help guide evolution. Artists can completely take over the role of a fitness function (Sims 1991; Todd and Latham 1999), and more recent work has investigated the use of these techniques for evolving photorealistic images of faces—potentially useful for the identification of criminals (Hancock and Frowd 1999, and see Chapter 16). These applications raise numerous human-computer interfacing issues: Will a creative system detrimentally affect the style of an artist or the memory of a crime victim? These software tools have been shown to aid imagination and creativity, but what is the best method to let the user inform evolution of his/her preferences, and what is the best method for the computer to report the structure and contents of the space being explored? Clearly, further research is required to address these issues.

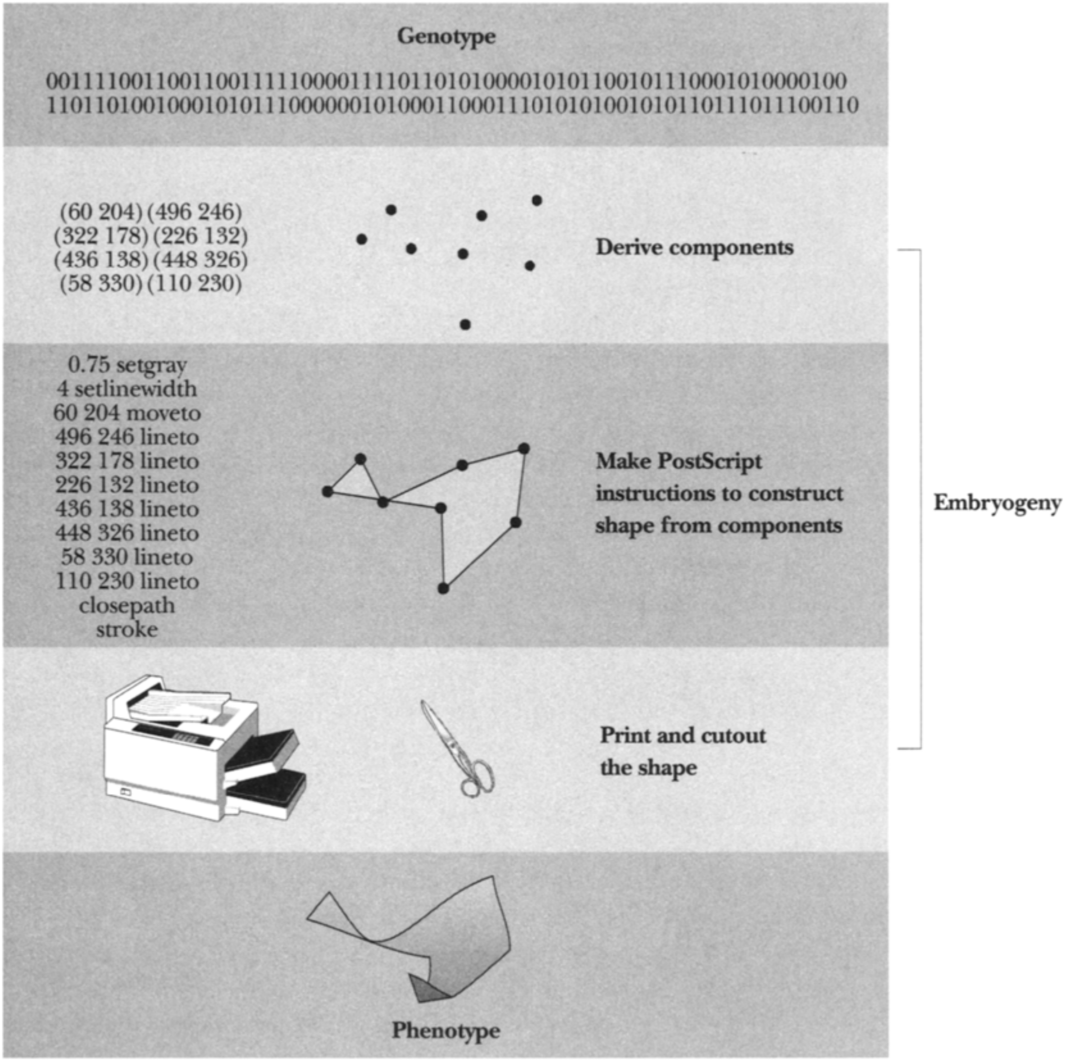
Example Creative Application

To illustrate the potential of creative evolutionary systems, we will now explore a simple application—the generation of shapes that, when constructed out of paper, fall as slowly as possible to the ground. In this case we will not use human interaction; this will be the second type of creative evolutionary system described earlier.

We can construct a simple creative evolutionary system, following the framework provided above. A "simple GA" can be used as the evolutionary algorithm (Goldberg 1989). The genotype representation can be a flat chromosome of 16 binary coded genes. A very basic mapping process or embryogeny can be used to derive 8 (x, y) parameter values from each chromosome, join each vector to its successor by an edge, join the last vector to the first with an edge, and fill the resulting shape. An easy way to perform this process is to execute PostScript instructions output by the system, print the shapes, and use scissors to cut out the shapes (see Figure I.18). The resulting paper phenotypes ("represented" by reality) can then be tested by releasing them from a height of 150 mm three times in succession. Each phenotype can be allocated their fitness score by calculating $1/(\text{time}_1 + \text{time}_2 + \text{time}_3)$, ensuring that evolution would attempt to generate phenotypes with increased "falling times."

This application illustrates the use of an extremely basic component-based embryogeny representation. As Figure I.18 illustrates, the components are simply eight vertices with (x, y) positions. Despite these components having no size and no type, the unconstrained freedom of position of each vertex relative to all other vertices means that this component-based representation allows the definition of a vast number of different shapes. Clearly this is a knowledge-lean representation (no information about which shapes are best is provided). It should also be clear that the representation allows evolution to explore the solution space in an unconstrained manner and that there will be many millions of good and bad solutions for this problem.

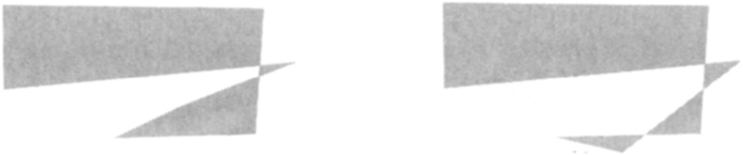
Predictably, the use of real-life testing is time-consuming and laborious (especially if you make the mistake of performing the experiment yourself instead of enlisting the services of a student). Consequently, for the purposes of this illustration, population sizes of 10 individuals were used in an evolutionary run of 10 generations.


Genotype
001111001100110011111000011110110101000010101100101110001010000100
1101101001000101011100000010100011000111010101001010110111011100110
(60 204) (496 246)
(322 178) (226 132)
(436 138) (448 326)
(58 330) (110 230)
Derive components
Make PostScript
instructions to construct
shape from components
Embryogeny
Print and cutout
the shape
Phenotype

```
0.75 setgray
4 setlinewidth
60 204 moveto
496 246 lineto
322 178 lineto
226 132 lineto
436 138 lineto
448 326 lineto
58 330 lineto
110 230 lineto
closepath
stroke
```

1.18
FIGURE

The paper fall application uses eight vertices as its components. These are extracted from the genotype and transformed into real paper shapes by the use of a "connect the dots and fill the shape" embryogeny (and someone to print and cut out the shape).



1.19
FIGURE
Two paper shapes evolved to fall slowly through the air. Both are members of the final generation, and both use a smaller "arm" or flap to cause the shape to rotate as it falls, like a sycamore seed. (Not shown at scale used for testing.)

The results were interesting. Despite these excessively low values, evolution was able to make significant improvements on the time taken for the shapes to reach the ground. In the experiments, times taken for initially random shapes to fall 150 mm varied from 0.7 seconds to 1.8 seconds. By the 10th generation, all shapes took, on average, more than 2 seconds to fall the same distance. Figure I.19 shows two of the solutions in the final population. Convergence has begun to occur, with most shapes using the same technique of having a smaller flap that causes the shapes to rotate as they fall. The main benefit of this solution appears to be the way rotation stabilizes the motion of the shape, preventing it from slipping sideways in the air and plummeting to the ground at tremendous speed. Even with the very limited resources evolution was given, a "creative" solution to the problem was found.

From Creative Evolutionary Systems to Creativity

This section of the introduction has introduced the ideas behind creative evolutionary systems. Such systems are used to aid the creativity of users by helping them to explore ideas during evolution, or help show users new ideas and concepts by generating innovative new solutions to problems previously thought to be only solvable by creative people. A brief background of the field was given, showing how difficult applications such as architecture, programming, circuit design, and art and music composition triggered the development of a new approach. By using human interaction we can enable the evolution of aesthetically pleasing and other difficult-to-evaluate solutions. By employing knowledge-lean component-based representations, we remove constraints to search in the representations and allow evolution to *assemble* new solutions. Using such methods we have turned evolution into an explorer of what is possible, instead of an optimizer of what is already there.

Now that we have a much clearer idea of what a creative evolutionary system is, it is time to look at the hardest question of all: Is it really creative?

IS EVOLUTION CREATIVE?

Although we have defined the term "creative evolutionary systems" earlier, this introduction would not be complete without some discussion of creativity itself. It is clear that evolution can aid in our own creativity, perform tasks that previously needed creative people, and, indeed, in some cases produce results in a seemingly creative manner. But is there any justification at all in using this rather contentious word to describe a scientific field? Some would argue yes, others no. This final section of the introduction explores the meaning of this word and how it relates to evolution.

Creativity

Creativity is somehow almost magical. We all recognize manifestations of creativity without effort, yet this mysterious property or ability remains aloof from our attempts to understand it. Creativity is another "fuzzy" word, a word with vast connotations, many inextricably entwined with the pride of individuals. It is a very "humanistic" word, used primarily to describe human skills and abilities, and almost without exception, it is used as a positive descriptor. And it is clear to see why the label "creative" is one to be proud of. With meanings such as *aesthetic, lovely, poetic, beautiful, skilled, proficient, inventive,* and *elegant,* surely this deceptively simple word is overflowing with compliments.

But not only does this word allude to admiration, it also implies genuine ability. Our most creative members of society are often regarded as our highest achievers. Whether the accomplishment is a new form of art that shocks in its radical novelty, or whether it is a theorem that describes the physical laws of our universe in a more concise and elegant manner, there can be little doubt of the creativity involved.

Yet creativity does not necessarily mean the creation of a tangible *something.* To be creative often appears to be the ability to find a solution where others fail—again, an extremely useful ability. It can be argued that many of the best leaders in history gained their successes through creative political (or military) thinking. Certainly, many of the remarkable feats of survival and rescue can be attributed to the quick thinking of creative individuals. A classic example is the recovery of the Apollo 13 crew from the brink of tragedy.

Being creative is clearly a good thing.

Evolution

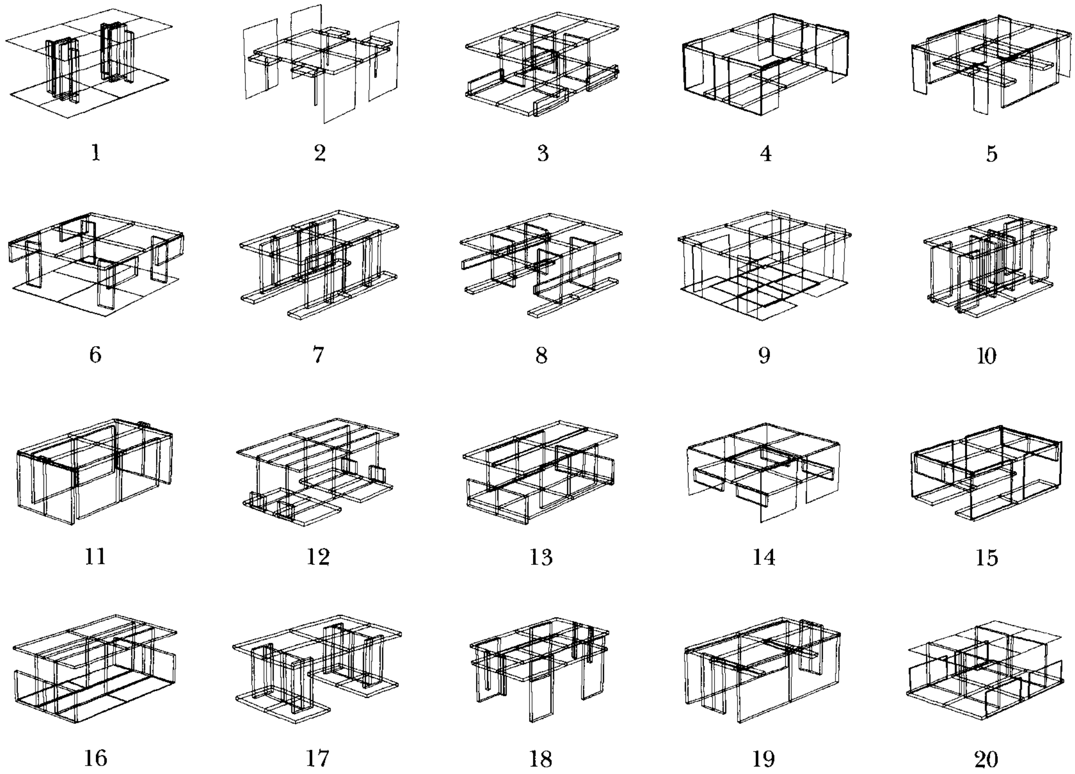
Evolution is not a person. It is an unthinking, blind process, a relentless procedure, a harsh and unconscious fact of life. How can we possibly call something so inhuman, so brutal, *creative?*

Perhaps we should not, but delving past such moral prejudices, evolution can be seen in a different light. Evolution has been hard at work creating the myriad forms of life that have lived and died on our world for billions of years. In that unimaginably vast amount of time, designs of life wholly beyond our current comprehension have emerged. From the complex miniature chemical factories contained within every cell of your body, to the immensely complicated organization of your brain, which even as you read this, performs unfathomable chemical and electrical changes, evolution is a master of design.

Examples of aesthetic, lovely, poetic, and beautiful evolved solutions surround us, are contained within us, and are us. Every living thing cries out proficiency, elegance, inventiveness, and skill in design. The abilities of natural evolution far surpass our most creative problem solvers. Moreover, as biologists uncover more information about the workings of the creatures around us, it is becoming clear that many "human" solutions have existed in nature long before they were thought of by any human (French 1994): for example, pumps, valves, heat exchange systems, optical lenses, sonar. Indeed, many of our recent designs borrow features directly from nature, such as the cross-sectional shape of aircraft wings from birds, and Velcro from certain types of "sticky" seeds.

Of course our own achievements are remarkable, and many do not exist in nature. Even something as simple as the wheel is not used in the natural world. But it must be remembered that natural evolution is constrained to the creation of life—all of its designs must be capable of self-replication, and nearly all must grow from a single cell, following the evolved instructions contained within the genetic makeup of that cell (Dawkins 1986).

But now this constraint is no longer valid. Evolutionary computation allows us to harness the power of evolution for nonliving designs. In this new digital domain, evolution can evolve the wheel or the electronic circuit. We can use evolution to generate music and art. Evolutionary algorithms permit us to exploit the remarkable properties of natural evolution, endowing our computers with skills that suspiciously resemble creativity. Even for such mundane tasks as evolving the design for a coffee table, evolution shows surprising originality. To illustrate this, Figure I.20 shows 20 coffee-table designs, evolved by Bentley's generic evolutionary design system (GADES). From the same set of functional criteria, and without any human intervention, 20 very different solutions were evolved (Bentley



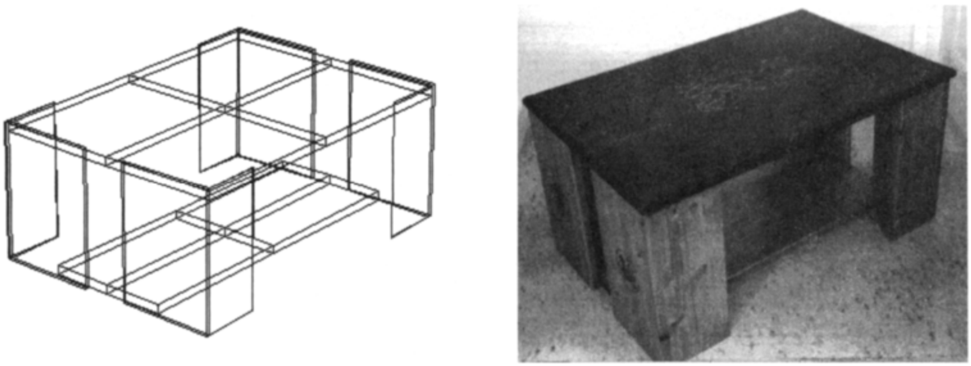
I.20 FIGURE Twenty coffee-table designs evolved by GADES.

1999a)—many using very original and unusual ideas. Figure I.21 shows the design that was ultimately chosen, and a photograph of the final table, built according to this design.

As the simple illustrative example shows, evolution certainly seems to exhibit some of the properties of creativity. It is still unclear, however, if evolution can truly be termed "creative."

Creative Evolution?

There are an almost unlimited number of different definitions that exist for creativity. To determine whether evolution can be considered creative, it seems appropriate to explore some of those definitions that deal with creativity in terms



(a) (b)

I.21
FIGURE

(a) The chosen coffee-table design (number 4) and (b) a photo of the actual table.

of computers and evolution. There are two main types of creativity that are considered here, similar to Gero's (1996) "cognitive" and "social" views: an individual can display creativity when performing some action, and the physical result of some action can display characteristics that may be regarded as being creative. The first two definitions we will consider refer to the results of evolution. They explore whether such evolved results can be considered creative, and if so, whether this implies evolution is creative. The next six definitions refer to the process itself—does evolution generate solutions to problems creatively?

Generating "surprising and innovative solutions" (Gero and Kazakov 1996). As many of the chapters of this book show, there can be no doubt that natural evolution is capable of innovation. It is also clear that evolutionary computation displays surprising levels of novelty. Even for very simple problems such as the one shown in Figure I.20, evolutionary algorithms are capable of finding unusual and original solutions.

Consequently, according to this definition, there can be no doubt that evolution is creative, but it seems that the definition may be too general. The patterns of frost on a window, snowflakes, sand dunes, and formations of eroded rock can all be described as surprising and innovative, so if evolution is called creative, then according to this interpretation, the laws of physics and the four elements must also be creative. Perhaps there is some justice to this, for if the blind process of evolution is creative, then why shouldn't the blind forces of nature such as tides and winds also have the same linguistic honor endowed upon them?

However, the difference between the evolution of life, and the erosion or formation of inanimate objects, seems too great to ignore.

Creating "novel solutions that are qualitatively better than previous solutions" (Gero and Kazakov 1996). This is a more rigorous definition, and it overcomes the problems discussed above. No snowflake or rock formation is better or worse than any other, they simply exist, and may or may not be elegant and attractive. In contrast, the essence of evolution is *improvement* over time. Evolution does generate qualitatively better solutions because, unlike inanimate objects in nature, evolution generates solutions better able to survive. Although the task of survival is constantly changing and the success rate of each living design constantly varies, useful survival skills such as the ability to fly, see, run, swim, and so on, have improved. Natural evolution, without doubt, generates qualitatively better solutions than previous ones.

In evolutionary computation, the same is true. Generation by generation, solutions are improved. Particularly in applications where evolution is permitted to vary aspects of the representation, the final evolved solutions are qualitatively better compared to the initially random solutions. When comparing designs evolved by computers with our own designs, evolution is also capable of providing substantial improvements and, in some cases, genuinely original design concepts (Bentley 1999a).

So evolution, once again, seems to be creative. But still the definition seems incomplete. Rather than focusing on the results of evolution and attempting to determine by proxy whether their generation implies creativity, it seems more appropriate to concentrate on *how* the solutions are found. Are they found *creatively?*

Less knowledge about existing relationships between the requirements and the form to satisfy those requirements (Rosenman 1997). This definition of creativity states that the ability to generate good solutions, even when very little or no information about the fundamental nature of the solutions is provided, implies that the generation process must be creative. Natural evolution always satisfies this, for there is no knowledge provided anywhere about which solutions should be favored—the fittest simply survive (unless one believes in divine intervention). Evolutionary computation does not always satisfy this definition. When using evolution to optimize given parameters in a predefined structure, considerable knowledge is embedded within that representation; hence there can be no creativity. However, for problems that employ evolution as a generative technique, such knowledge can be reduced to a bare minimum (e.g., a design grammar that provides a means to define designs without indirectly providing knowledge of which designs are best). Because of this greater discrimination, Bentley employs this definition of creativity for *creative evolutionary design* (Bentley 1999c). However, some regard this as insufficient to capture the essence of creativity.

Exploring a search space in an innovative and efficient way. From a computational point of view, evolution is simply a special kind of search algorithm. Some argue that for evolution to be considered creative, it must traverse its search spaces in a creative manner; that is, it must be innovative or efficient in its search. Exhaustive search and random search are examples of noncreative techniques. Evolutionary algorithms are good examples of creative search. Although we have few proofs that coherently describe the behavior of evolutionary algorithms, through experimentation and analysis we have learned that evolutionary techniques have excellent abilities as general-purpose problem solvers. Indeed, as Goldberg (1989) states, the genetic algorithm is "a search algorithm with some of the innovative flair of human search."

In a sense, this definition draws a parallel between the innovative thinking by a creative person, and the innovative searching by a creative algorithm. However, the application of this definition remains difficult, for although it seems that evolution probably is creative, it would seem to be just as hard to define the boundaries between creative and noncreative search as it is to define them between creative and noncreative thought.

Exploring alternative search spaces (Gero 1996). By redefining the search space, or indeed, constructing new search spaces in which to find solutions, a search process can be considered creative according to this definition. Just as our creative thinkers find alternative ways to look at problems, if evolution can enhance or change its search space, it will be creative.

This is another, more discriminative interpretation, which is a little harder for evolution to satisfy. Using evolution as a simple optimizer of fixed parameters is clearly not creative. However, natural evolution and some of the more advanced evolutionary algorithms are capable of varying their representations. Such evolutionary approaches can have considerable freedom to modify their representations in parallel to the evolution of solutions. They can alter the coding, vary the genome length, employ redundant genetic material, select useful functional elements, and even create higher-level building blocks. Once again, at least some of the more complex forms of evolution can be considered creative.

Transferring useful information from other domains (Goldberg 1999; Holland 1998). Goldberg (1999) makes the distinction between innovation and creativity with this definition. He feels that innovation involves discovery *within* a discipline, whereas creativity requires a transfer of knowledge from *without.* Holland (1998) makes a very similar point in his discussion of metaphors, and how knowledge in one area can be applied to a different subject in order to change the perceptions and understanding of that subject.

In nature there are no clearly defined domains of knowledge. Perhaps different species could be regarded as distinct archives of knowledge, but natural evolution does not transfer such information, for interbreeding is usually unsuccessful. Some argue that knowledge of previous solutions that were successful during alternative environmental conditions is held in junk DNA for future reuse, but this is hardly knowledge transfer from a different domain. Goldberg does suggest that the transfer of knowledge about better knowledge representation may be one aspect of creativity, and it is argued that natural evolution does evolve such evolvability (Dawkins 1989). However, this seems a somewhat contrived argument. Consequently, according to this definition, natural evolution is probably not creative.

Evolutionary computation does not currently include methods to identify and transfer knowledge from other domains to aid search. Goldberg feels that it is conceivable in the future, but it seems unlikely that evolution will be used to perform this; rather the "creative" process according to this definition would be performed by the knowledge identification, transfer, and conversion software.

Going beyond the bounds of a representation (Boden 1992). Boden (1992) feels that to be creative it is necessary to find a novel solution that simply could not have been defined by a representation. She suggests that this is the nature of a paradigm shift, where entirely new approaches to the representation of problems are found. This precludes the transfer of knowledge into the current representation as suggested by the previous definition. Instead, a different representation would be required. Boden does not feel that computers will ever be capable of such creativity (Boden 1992).

Although this may appear analogous to the behavior of creative thinkers in our society, it seems to ignore the fact that our own brains are fundamentally single-representation devices. At the lowest level, they must always use neurons and chemical and electrical signals, so although many alternative higher-level representations can be expressed, they must always be defined using this "wetware."

Evolution (natural and computational) is similarly constrained to a low-level representation, this time genetic, but equally capable of defining higher-level representations of immense diversity and complexity. It therefore seems that, with respect to this definition, evolution and the human brain are equally likely (or unlikely) to be capable of creativity.

Expressing your soul. Some insist that creativity is an expression of your soul. This is one of the more controversial definitions of creativity and one of the hardest to satisfy for nonhuman activity. Whether you believe in the soul or not, few would argue that a computer or the process of evolution possesses one. Clearly, evolutionary computation cannot satisfy this definition—unless we ever construct living machines, whatever they might be. However, natural evolution is a special case—for if it is the working of God, then it must be creative.

Subjective Creativity

According to six (out of eight) of the definitions we have examined here, the more advanced forms of evolution can be considered to be creative. But in the end, it seems that the magical property of creativity is simply too subjective and anthropocentric a word to allow its unequivocal usage for evolution. The final judgment must be a personal one, but it is possible to give four responses from people with different outlooks on the world. These views are amalgamations of the views presented during a number of discussions on this subject. Although titles for each type of view have been given, this is more for convenience of reference than an attempt at accurate generalization.

The Atheist does not believe in creativity. All novelty is a result of random chance; complexity is a result of physical laws and natural selection. "Creative" individuals are simply individuals with brains that are better able to process information.

The Religious believe if natural evolution is considered to be the way in which God designs the living world, then it must be creative. However, a computer performing evolution does not involve God in this way, or a human soul, so it cannot be creative.

The Artist, often inspired and awed by the forms evolved in nature, finds evolution to be very creative.

The Scientist finds the results of evolution to be creative, but expresses doubts as to whether the process can be given the human descriptive word "creative."

CONCLUSIONS

We have now finished our introduction to creative evolutionary systems. We began with an overview of evolutionary algorithms explaining the main algorithms, and showing how all evolutionary algorithms are fundamentally the same. We then defined and described creative evolutionary systems, showing why they were developed and how user interaction and changes of representation can expand the capabilities of evolution. The last section of the introduction explored the taxing question of whether a creative evolutionary system can be said to actually work creatively.

But this was a mere introduction to the diversity of techniques that fall under the heading of creative evolutionary systems. It was intended to provide you with some grounding, making your explorations of the chapters in this book more fruitful. And we hope you do find this book useful. As the first of its kind on this topic, we feel confident that it provides the latest and most up-to-date ideas, methods, and results for creative evolutionary systems.

ACKNOWLEDGMENTS

Thanks to Tina Yu, who provided help, time, and some of the text. Thanks to Suran Goonatilake and Phil Treleaven for their advice. Thanks to Jonathan Wakefield, Sanjeev Kumar, and all the members of UCL's Design Group, and to Martin Oates, Joshua Knowles, and other members of Reading University's PEDAL Lab and the Artificial Symbiosis project team, for providing stimulating discussions and helpful criticism. Also thanks to Bill Punch for the "innovative and efficient" idea, and to the director at Fulcrum Productions for the "souls" idea. Thanks also to the other members of the J13 team, the guys in 301 and 303, and the members of UCL's Design Group and nUCLEAR who provided their views for this introduction.

RECOMMENDED READINGS

AI and Creativity

Artificial Intelligence, by Elaine Rich and Kevin Knight (1991).

Artificial Intelligence: A Modern Approach, by Start Russell and Peter Norvig (1994).

Artificial Minds, by Stan Franklin (1997).

Dimensions of Creativity, edited by Margaret Boden (1996).

Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine, by Selmer Bringsjord and David Ferrucci (1999).

Genetic Algorithms

Adaptation in Natural and Artificial Systems, by John Holland (1975).

Genetic Algorithms in Search, Optimization and Machine Learning, by David Goldberg (1989).

The Handbook of Genetic Algorithms, edited by Lawrence Davis (1991).

Genetic Algorithms + Data Structures = Evolution Programs, by Zbigniew Michalewicz (1996).

Practical Handbook of Genetic Algorithms, edited by Lance Chambers (1995).

An Introduction to Genetic Algorithms, by Melanie Mitchell (1996).

Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, 2nd ed., by David B. Fogel (2000).

The Design of Innovation: Lessons from Genetic Algorithms, by David Goldberg (2000).

Genetic Programming

Genetic Programming: On the Programming of Computers by Means of Natural Selection, by John Koza (1992).

Genetic Programming II: Automatic Discovery of Reusable Programs, by John Koza (1994).

Genetic Programming III, by John Koza et al. (1999).

Advances in Genetic Programming, edited by Kenneth E. Kinnear, Jr. (1994).

Advances in Genetic Programming 2, edited by Peter Angeline and Kenneth E. Kinnear, Jr. (1996).

Genetic Programming—An Introduction, by Wolfgang Banzhaf et al. (1998).

Data Structures and Genetic Programming, by Bill Langdon (1998).

Evolution Strategies

Evolutionsstrategie: Optimierung Technischer Systeme nach Prinzipien der Biologischen Evolution, by Ingo Rechenberg (1973).

Numerical Optimization of Computer Models, by H.-P. Schwefel (1981).

Evolution and Optimum Seeking, by H.-P. Schwefel (1995).

Evolutionsstrategie '94 (volume 1 of *Werkstatt Bionik und Evolutionstechnik*), by Ingo Rechenberg (1994).

Evolutionary Algorithms in Theory and Practice, by Thomas Bäck (1996).

Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, by David B. Fogel (2000).

Evolutionary Programming

Artificial Intelligence through Simulated Evolution, by Lawrence Fogel et al. (1966).

System Identification through Simulated Evolution: A Machine Learning Approach to Modeling, by David B. Fogel (1991).

Evolutionary Algorithms in Theory and Practice, by Thomas Bäck (1996).

Intelligence through Simulated Evolution, by Lawrence Fogel (1999).

Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, by David B. Fogel (2000).

REFERENCES

Adeli, H., and N. T. Cheng. (1994). Concurrent genetic algorithms for optimization of large structure. *Journal of Aerospace Engineering* 7(3):276–296.

Altenberg, L. (1995). The Schema Theorem and Price's Theorem. In L. D. Whitley and M. D. Vose (eds.), *Foundations of Genetic Algorithms 3,* Morgan Kaufmann, pp. 23–49.

Angeline, P., and K. E. Kinnear Jr. (eds.) (1996). *Advances in Genetic Programming 2.* MIT Press.

Axelrod, R. (1987). The Evolution of Strategies in the Iterated Prisoner's Dilemma. In L. Davis (ed.), *Genetic Algorithms and Simulated Annealing,* Pitman, pp. 32–41.

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press.

Baluja, S., and R. Caruna (1995). Removing the Genetics from the Standard Genetic Algorithm. In A. Prieditis and S. Russell (eds.), *Proceedings of the Twelfth International Conference on Machine Learning,* ML-95, pp. 38–46.

Banzhaf, W. (1994). Genotype-Phenotype-Mapping and Neutral Variation—A Case Study in Genetic Programming. In Y. Davidor, H-P Schwefel, and R. Mnner (eds.), *Parallel Problem Solving from Nature,* 3, Springer-Verlag, pp. 322–332.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone (1998). *Genetic Programming—An Introduction.* Morgan Kaufmann.

Bentley, P. J. (1996). *Generic Evolutionary Design of Solid Objects Using a Genetic Algorithm.* Ph.D. Thesis, Division of Computing and Control Systems, Department of Engineering, University of Huddersfield.

Bentley, P. J. (1997). The Revolution of Evolution for Real-World Applications. *Emerging Technologies '97: Theory and Application of Evolutionary Computation.* University College London.

Bentley, P. J. (1998a). Aspects of Evolutionary Design by Computers. In *Proceedings of the 3rd On-line World Conference on Soft Computing in Engineering Design and Manufacturing* (WSC3).

Bentley, P. J. (ed.) (1998b). *Proc. of the Workshop on Evolutionary Design,* 5th International Conference on Artificial Intelligence in Design '98, Instituto Superior Técnico Lisbon, Portugal, July 20–23, 1998.

Bentley, P. J. (ed.) (1999a). *Evolutionary Design by Computers.* Morgan Kaufmann.

Bentley, P. J. (1999b). Is Evolution Creative? In P. J. Bentley and D. Corne (eds.), *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems (CES),* Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB), pp. 28–34.

Bentley, P. J. (1999c). An Introduction to Evolutionary Design by Computers. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann, pp. 1–73.

Bentley, P. J., and D. Corne (eds.) (1999). *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems (CES).* Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB).

Bentley, P. J., and S. Kumar (1999). Three Ways to Grow Designs: A Comparison of Embryogenies for an Evolutionary Design Problem. In *Genetic and Evolutionary Computation Conference (GECCO '99),* July 14–17, 1999, Orlando, Florida, pp. 35–43.

Bentley, P. J., and J. P. Wakefield, (1995). The Evolution of Solid Object Designs Using Genetic Algorithms. In *Applied Decision Technologies (ADT '95),* April, pp. 391–400.

Bentley, P. J., and J. P. Wakefield (1996a). Generic Representation of Solid Geometry for Genetic Search. *Microcomputers in Civil Engineering* 11(3):153–161.

Bentley, P. J., and J. P. Wakefield (1996b). The Evolution of Solid Object Designs Using Genetic Algorithms. In V. Rayward-Smith (ed.), *Modern Heuristic Search Methods,* John Wiley, pp. 199–215.

Bentley, P. J., and J. P. Wakefield (1996c). Hierarchical Crossover in Genetic Algorithms. In *Proceedings of the 1st On-line Workshop on Soft Computing* (WSC1), Nagoya University, Japan, pp. 37–42.

Bentley, P. J., and J. P. Wakefield (1997a). Conceptual Evolutionary Design by Genetic Algorithms. *Engineering Design and Automation Journal* 3(2):119–131.

Bentley, P. J., and J. P. Wakefield (1997b). Generic Evolutionary Design. In P. K. Chawdhry, R. Roy, and R. K. Pant (eds.), *Soft Computing in Engineering Design and Manufacturing,* Springer-Verlag, Part 6, pp. 289–298.

Bentley, P. J., and J. P. Wakefield (1997c). Finding Acceptable Solutions in the Pareto-Optimal Range Using Multiobjective Genetic Algorithms. In P. K. Chawdhry, R. Roy, and R. K. Pant (eds.), *Soft Computing in Engineering Design and Manufacturing,* Springer-Verlag, Part 5, pp. 231–240.

Boden, M. A. (1992). *The Creative Mind: Myths and Mechanisms.* Basic Books.

Boden, M. A. (ed.) (1996). *Dimensions of Creativity.* MIT Press.

Born, J. (1978). *Evolutionsstrategien zur numerischen Lösung von Adaption-saufgaben.* Dissertation A, Humboldt-Universität, Berlin.

Bossert, W. (1967). Mathematical Optimization: Are There Abstract Limits on Natural Selection? In P. S. Moorhead and M. M. Kaplan (eds.), *Mathematical Challenges to the Neo-Darwinian Interpretation of Evolution,* Wistar Institute Press, pp. 35–46.

Box, G. E. P. (1957). Evolutionary operation: A Method for Increasing Industrial Productivity, *Applied Statistics* 6(2):81–101.

Bremmerman, H. J. (1958). The Evolution of Intelligence. The Nervous System as a Model of Its Environment. Technical Report No. 1. Contract No. 477(17). Department of Mathematics, University of Washington, Seattle.

Bremmerman, H. J. (1962). Optimization through Evolution and Recombination. In M. C. Yovits, G. T. Jacobi, and G. D. Goldstein (eds.), *Self-Organizing Systems,* Spartan Books, pp. 93–106.

Bringsjord, S., and D. Ferrucci (1999). *Artificial Intelligence and Literary Creativity: Inside the Mind of Brutus, a Storytelling Machine.* Lawrence Erlbaum Associates.

Chambers, L. (1995). *Practical Handbook of Genetic Algorithms.* CRC Press.

Chellapilla, K. (1997). Evolutionary Programming with Tree Mutations: Evolving Computer Programs without Crossover. In J. Koza et al. (eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference,* Morgan Kaufmann.

Chellapilla, K. (1998). Combining Mutation Operators in Evolutionary Programming. *IEEE Transactions on Evolutionary Computation* 2(3):91–96.

Chellapilla, K., and D. B. Fogel (1999). Evolving Neural Networks to Play Checkers Without Relying on Expert Knowledge. *IEEE Transactions on Neural Networks* 10(6):1382–1391.

Coates, P. (1997). Using Genetic Programming and L-Systems to Explore 3D Design Worlds. In R. Junge (ed.), *CAADFutures'97,* Kluwer Academic.

Collins, R. J., and D. R. Jefferson (1991). Selection in Massively Parallel Genetic Algorithms. In *Proceedings of the 4th International Conference on Genetic Algorithms,* pp. 249–256.

Corne, D., M. Dorigo, and F. Glover (eds.) (1999). *New Ideas in Optimization.* McGraw-Hill.

Corne, D., M. Oates, and G. D. Smith (eds.) (2000). *Telecommunications Optimization: Heuristic and Adaptive Techniques.* Wiley.

Cramer, N. L. (1985). Representation for the Adaptive Generation of Simple Sequential Programs. In J. J. Grefenstette (ed.), *Proceedings of an International Conference on Genetic Algorithms and Their Applications,* Lawrence Erlbaum, pp. 183–187.

Dasgupta, D., and D. R. McGregor (1992). Nonstationary Function Optimization Using the Structured Genetic Algorithm. In *Parallel Problem Solving from Nature 2,* Elsevier Science, pp. 145–154.

Dasgupta, D., and Z. Michalewicz (1997). *Evolutionary Algorithms in Engineering Applications.* Springer-Verlag.

Davidor, Y. (1991). A Naturally Occuring Niche and Species Phenomenon: The Model and First Results. In *Proceedings of the 4th International Conference on Genetic Algorithms,* pp. 257–263.

Davis, L. (1991). *The Handbook of Genetic Algorithms.* Van Nostrand Reinhold.

Dawkins, R. (1976). *The Selfish Gene.* Oxford University Press.

Dawkins, R. (1982). *The Extended Phenotype.* Oxford University Press.

Dawkins, R. (1983). Universal Darwinism. In D. Bendall (ed.), *Evolution from Molecules to Men,* Cambridge University Press.

Dawkins, R. (1986). *The Blind Watchmaker.* Longman Scientific and Technical.

Dawkins, R. (1989). The Evolution of Evolvability. In C. G. Langton (ed.), *Artificial Life. The Proceedings of an Interdisciplinary Workshop on the Synthesis and Simulation of Living Systems,* Vol. VI, September 1987, Los Alamos, New Mexico, Addison-Wesley, pp. 201–220.

Dawkins, R. (1996). *Climbing Mount Improbable.* Penguin Books.

Deb, K. (1991). *Binary and Floating Point Function Optimization Using Messy Genetic Algorithms.* Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 91004.

Deb, K., and D. E. Goldberg (1991). *mGA in C: A Messy Genetic Algorithm in C.* Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 91008.

Deb, K., and D. E. Goldberg (1993). Analyzing Deception in Trap Functions. In D. Whitley (ed.), *Foundations of Genetic Algorithms 2,* Morgan Kaufmann.

Deb, K., J. Horn, and D. E. Goldberg (1993). Multimodal Deceptive Functions. *Complex Systems* 7(2):131–153.

De Jong, K. A. (1975). *An Analysis of the Behaviour of a Class of Genetic Adaptive Systems.* Doctoral dissertation, University of Michigan. Dissertation Abstracts International.

Ebcioglu, K. (1988). An Expert System for Harmonizing Four-part Chorales. *Computer Music Journal* 12(3):43–51.

Eby, D., R. Averill, B. Gelfand, W. Punch, O. Mathews, and E. Goodman (1997). An Injection Island GA for Flywheel Design Optimization. In *5th European Congress on Intelligent Techniques and Soft Computing EUFIT '97,* vol. 1, pp. 687–691.

Eshelman, L. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In G. J. G. Rawlins (ed.), *Foundations of Genetic Algorithms,* Morgan Kaufmann.

Flemming, U. (1987). More than the Sum of Parts: The Grammar of Queen Anne Houses. *Environment and Planning B* 14:323–350.

Fogel, D. (1992a). *Evolving Artificial Intelligence.* PhD thesis, University of California, San Diego.

Fogel, D. (1992b). An analysis of evolutionary programming. *Proc. of the 1st Annual Conf. on Evolutionary Programming,* San Diego, pp. 43–51.

Fogel, D. B. (1991). *System Identification through Simulated Evolution: A Machine Learning Approach to Modeling.* Ginn.

Fogel, D. B. (1994). Asymptotic Convergence Properties of Genetic Algorithms and Evolutionary Programming: Analysis and Experiments. *Journal of Cybernetics and Systems* 25:389–407.

Fogel, D. B. (1997). The Advantages of Evolutionary Computation. In D. Lundth, B. Olsson, and A. Naraganan (eds.), *Biocomputing and Emergent Computation 1997,* World Scientific Press, pp. 1–11.

Fogel, D. B. (ed.) (1998). *Evolutionary Computation: The Fossil Record.* IEEE Press.

Fogel, D. B. (2000). *Evolutionary Computation: Towards a New Philosophy of Machine Intelligence.* Second edition. IEEE Press.

Fogel, G. B., and D. B. Fogel (1995). Continuous Evolutionary Programming: Analysis and Experiments. *Journal of Cybernetics and Systems* 26:79–90.

Fogel, L. J. (1963). *Biotechnology: Concepts and Applications.* Prentice-Hall.

Fogel, L. J. (1999). *Intelligence through Simulated Evolution.* John Wiley.

Fogel, L. J., P. Angeline, and D. B. Fogel (1995). An Evolutionary Programming Approach to Self-Adaptation on Finite State Machines. In J. R. McDonnell, R. G. Reynolds, and D. B. Fogel (eds), *Proceedings of the Fourth International Conference on Evolutionary Programming,* MIT Press.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution.* Wiley.

Fonseca, C. D. M., and P. J. Fleming (1995). An Overview of Evolutionary Algorithms in Multiobjective Optimization. *Evolutionary Computation* 3(1):1–16.

Forrest, S., A. S. Perelson, L. Allen, and R. Cherukuri (1995). A Change-Detection Algorithm Inspired by the Immune System. *IEEE Transactions on Software Engineering.*

Franklin, S. (1997). *Artificial Minds.* MIT Press.

Fraser, A. S. (1957a). Simulation of Genetic Systems by Automatic Digital Computers. I. Introduction. *Australian Journal of Biological Sciences* 10:484–491.

Fraser, A. S. (1957b). Simulation of Genetic Systems by Automatic Digital Computers. II. Effects of Linkage or Rates of Advance under Selection. *Australian Journal of Biological Sciences* 10:492–499.

Frazer, J. (1995). *An Evolutionary Architecture.* Architectural Association, London.

Friedberg, R. M. (1958). A Learning Machine: Part I. *IBM Journal of Research and Development* 2(1):2–13.

Friedman, G. J. (1956). *Selective Feedback Computers for Engineering Synthesis and Nervous System Analogy.* Master's thesis, UCLA.

French, M. J. (1994). *Invention and Evolution: Design in Nature and Engineering.* Second edition. Cambridge University Press.

French, M. J. (1999). The Interplay of Evolution and Insight in Design. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

French, M., and A. C. Ramirez (1996). Toward A Comparative Study of Quarter-Turn Pneumatic Valve Actuators. *Journal of Engineering Manufacture,* part B, 543–552.

Funes, P., and J. Pollack (1997). *Computer Evolution of Buildable Objects.* Brandeis University Computer Science Technical Report CS-97-191.

Funes, P., and J. Pollack (1999). The Evolution of Buildable Objects. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

Furuta, H., K. Maeda, and W. Watanabe (1995). Application of Genetic Algorithm to Aesthetic Design of Bridge Structures. *Microcomputers in Civil Engineering* 10(6):415–421.

Gehlhaar, D. K., and D. B. Fogel (1996). Tuning Evolutionary Programming for Conformationally Flexible Molecular Docking. In L. Fogel, P. Angeline, and T. Bäck (eds.), *Proceedings of the Fifth International Conference on Evolutionary Programming,* MIT Press.

Gero, J. S. (1996). Computers and Creative Design. In M. Tan and R. Teh (eds.), *The Global Design Studio,* National University of Singapore, pp. 11–19.

Gero, J. S., and V. Kazakov (1996). An Exploration-Based Evolutionary Model of Generative Design Process. *Microcomputers in Civil Engineering* 11:209–216.

Gero, J. S., and M. L. Maher (eds.) (1993). *Modeling Creativity and Knowledge-Based Creative Design.* Lawrence Erlbaum.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Goldberg, D. E. (1991). Genetic Algorithms as a Computational Theory of Conceptual Design. In *Proc. of Applications of Artificial Intelligence in Engineering 6,* pp. 3–16.

Goldberg, D. E. (1994). Genetic and Evolutionary Algorithms Come of Age. *Communication of the ACM* 37(3):113–119.

Goldberg, D. E. (1999). The Race, the Hurdle, and the Sweet Spot: Lessons from Genetic Algorithms for the Automation of Design Innovation and Creativity. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

Goldberg, D. (2000). *The Design of Innovation: Lessons from Genetic Algorithms.* (In press).

Goldberg, D. E., et al. (1992). Accounting for Noise in the Sizing of Populations. In *Foundations of Genetic Algorithms 2,* Morgan Kaufmann, pp. 127–140.

Greffenstette, J. J. (1991). Strategy Acquisition with Genetic Algorithms. In *The Handbook of Genetic Algorithms,* Van Nostrand Reinhold, pp. 186–201.

Hancock, P., and C. Frowd (1999). Evolutionary Generation of Faces. In P. J. Bentley and D. W. Corne (eds.), *Proceedings of the AISB'99 Symposium on Creative Evolutionary Systems (CES),* Society for the Study of Artificial Intelligence and Simulation of Behaviour (AISB).

Harris, R. A. (1994). An Alternative Description to the Action of Crossover. In *Proceedings of Adaptive Computing in Engineering Design and Control '94,* pp. 151–156.

Harvey, I. (1992). The SAGA Cross: The Mechanics of Recombination for Species with Variable-Length Genotypes. In R. Manner and B. Manderick (eds.), *Parallel Problem Solving from Nature II,* pp. 269–278.

Harvey, I. (1997). Cognition Is Not Computation: Evolution Is Not Optimization. In W. Gerstner, A. Germond, M. Hasler, and J.-D. Nicoud (eds.), *Artificial Neural Networks—ICANN97, Proc. of 7th International Conference on Artificial Neural Networks,* October 7–10, 1997, Lausanne, Switzerland, Springer-Verlag, LNCS 1327, pp. 685–690.

Harvey, I., and A. Thompson (1997). Through the Labyrinth Evolution Finds a Way: A Slicon Ridge. In T. Higuchi and M. Iwata (eds.), *Proceedings of the 1st Int. Conf. on Evolveable Systems: From Biology to Hardware (ICES96),* Springer-Verlag, LNCS 1259, pp. 406–422.

Holland, J. H. (1973). Genetic Algorithms and the Optimal Allocations of Trials. *SIAM Journal of Computing* 2(2):88–105.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Holland, J. H. (1992). Genetic Algorithms. *Scientific American* July, 66–72.

Holland, J. H. (1998). *Emergence: Chaos to Order.* Oxford University Press.

Horn, J. (1993). Finite Markov Chain Analysis of Genetic Algorithms with Niching. In *Proceedings of the Fifth International Conference on Genetic Algorithms,* Morgan Kaufmann, pp. 110–117.

Horn, J., and N. Nafpliotis (1993). *Multiobjective Optimization Using the Niched Pareto Genetic Algorithm.* Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 93005.

Horn, J., et al. (1994). *Implicit Niching in a Learning Classifier System: Nature's Way.* Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 94001.

Husbands, P., G. Jermy, M. McIlhagga, and R. Ives (1996). Two Applications of Genetic Algorithms to Component Design. In T. Fogarty (ed.), *Selected Papers from AISB Workshop on Evolutionary Computing,* Springer-Verlag, Lecture Notes in Computer Science, pp. 50–61.

Jakobi, N. (1996). Harnessing Morphogenesis. In *Proceedings of the International Conference on Information Processing in Cell and Tissue.*

Kanal, L., and V. Cumar (eds.) (1988). *Search in Artificial Intelligence.* Spring-Verlag.

Kargupta, H. (1993). *Information Transmission in Genetic Algorithm and Shannon's Second Theorem.* Illinois Genetic Algorithms Laboratory (IlliGAL), report no. 93003.

Kinnear, K. E., Jr. (ed.) (1994). *Advances in Genetic Programming.* MIT Press.

Knight, T. W. (1980). The Generation of Hepplewhite-Style Chair-Back Designs. *Environment and Planning B* 7:227–238.

Knowles, J. D., and D. W. Corne (2000). Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation* 8(2):149–172.

Koning, H., and J. Eizenberg (1981). The Language of the Prairie: Frank Lloyd Wright's Prairie Houses. *Environment and Planning B* 8:295–323.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Koza, J. (1994). *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press.

Koza, J. (1999). 1000-Pentium Beowulf Computer for Genetic Programming Research. Message posted to *Evolutionary Computing Mailbase List,* August 10, 1999.

Koza, J. R., F. R. Bennett III, D. Andre, and M. A. Keane (1999). *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann.

Langdon, B. (1998). *Data Structures and Genetic Programming: Genetic Programming + Data Structures = Automatic Programming!* Kluwer.

Langdon, B., and R. Poli (1997). Fitness Causes Bloat. In P. K. Chawdhry, R. Roy, and R. K. Pant (eds.), *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2),* London, June 23–27, 1997, Springer-Verlag.

Langton, C. (ed.) (1995). *Artificial Life: An Overview.* MIT Press.

Levine, D. (1994). *A Parallel Genetic Algorithm for the Set Partitioning Problem.* Ph. D. dissertation, Argonne National Laboratory, Illinois.

Lohn, J., and J. Reggia (1995). Discovery of Self-Replicating Structures Using a Genetic Algorithm. In *1995 IEEE Int. Conf. on Evolutionary Computation (ICEC '95),* Perth, Western Australia, vol. 1, pp. 678–683.

Lund, H., L. Pagliarini, and O. Miglino (1995). Artistic Design with GA and NN. In *Proc. of the 1st Nordic Workshop on Genetic Algorithms and Their Applications (1NWGA),* Uni. Vaasa, Finland, pp. 97–105.

Maynard Smith, J. (1989). *Evolutionary Genetics.* Oxford University Press.

Michalewicz, Z. (1996). *Genetic Algorithms + Data Structures = Evolution Programs.* Third edition. Springer.

Mitchell, M. (1996). *An Introduction to Genetic Algorithms.* MIT Press.

Moscato, P. (1999). Memetic Algorithms: A Short Introduction. In D. Corne, M. Dorigo, and F. Glover (eds.), *New Ideas in Optimization,* McGraw-Hill, pp. 219–234.

O'Reilly, U.-M., and F. Oppacher (1995). The Troubling Aspects of a Building Block Hypothesis for Genetic Programming. In L. D. Witley and M. D. Vose (eds.), *Foundations of Genetic Algorithms,* Morgan Kaufmann, pp. 72–88.

Pachet, F., G. Ramalho, and J. Carrive (1996). Representing Temporal Musical Objects and Reasoning in the MusES System. *Journal of New Music Research* 25(3):252–275.

Page, J., R. Poli, and W. Langdon (1999). Smooth Uniform Crossover with Smooth Point Mutation in Genetic Programming: A Preliminary Study. In R. Poli, P. Nordin, W. B. Langdon, and T. Fogarty (eds.), *Proceedings of the Second European Workshop on Genetic Programming—EuroGP'99,* Goteborg, May 26–27, 1999, Springer-Verlag.

Parmee, I. (1996). Towards an Optimal Engineering Design Process Using Appropriate Adaptive Search Strategies. *Journal of Engineering Design* 7:4.

Parmee, I. (1999). Exploring the Design Potential of Evolutionary Search, Exploration and Optimization. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

Parmee, I. C., and M. J. Denham (1994). The Integration of Adaptive Search Techniques with Current Engineering Design Practice. In *Proc. of Adaptive Computing in Engineering Design and Control '94,* Plymouth, pp. 1–13.

Paton, R. (1994). Enhancing Evolutionary Computation Using Analogues of Biological Mechanisms. In *Evolutionary Computing, AISB Workshop,* Springer-Verlag, pp. 51–64.

Poli, R., and Langdon B. (1997a). Genetic Programming with One-Point Crossover. In P. K. Chawdhry, R. Roy, and R. K. Pant (eds.), *Second On-line World Conference on Soft Computing in Engineering Design and Manufacturing (WSC2),* London, June 23–27, 1997, Springer-Verlag.

Poli, R., and B. Langdon (1997b). A New Schema Theorem for Genetic Programming with One-Point Crossover and Point Mutation. In J. Koza et al. (eds.), *Genetic Programming 1997: Proceedings of the Second Annual Conference on Genetic Programming,* Morgan Kaufmann, pp. 278–285.

Price, G. R. (1970). Selection and Covariance. *Nature* 227(1):520–521.

Radcliffe, N. J., and P. D. Surry (1994a). Formal Memetic Algorithms. *Edinburgh Parallel Computing Centre.*

Radcliffe, N. J., and P. D. Surry (1994b). Co-operation through Hierarchical Competition in Genetic Data Mining. Submitted to *Parallel Problem Solving from Nature.*

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution.* Fromman-Holzboog Verlag.

Rechenberg, I. (1994). *Evolutionsstrategie '94,* volume 1 of *Werkstatt Bionik und Evolutionstechnik.* Frommann-Holzboog.

Rich, E., and K. Knight (1991). *Artificial Intelligence.* McGraw-Hill.

Rosenman, M. (1997). The Generation of Form Using an Evolutionary Approach. In D. Dasgupta and Z. Michalewicz (eds.), *Evolutionary Algorithms in Engineering Applications,* Springer-Verlag, pp. 69–86.

Rowbottom, A. (1999). Evolutionary Art and Form. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

Rudolph, G. (1996). Convergence of Evolutionary Algorithms in General Search Spaces. In *Proceedings of the Third IEEE Conference on Evolutionary Computation,* IEEE Press, pp. 50–54.

Rudolph, G. (1997). Convergence Rates of Evolutionary Algorithms for a Class of Convex Objective Functions. *Control and Cybernetics* 26(3):375–390.

Rudolph, G. (1998). Local Convergence Rates of Simple Evolutionary Algorithms with Cauchy Mutations. *IEEE Transactions on Evolutionary Computation* 1(4).

Russell, S., and P. Norvig (1994). *Artificial Intelligence: A Modern Approach.* Prentice-Hall.

Schaffer, J. D. (1985). Multiple Objective Optimization with Vector Evaluated Genetic Algorithms. *Genetic Algorithms and Their Applications: Proceedings of the First International Conference on Genetic Algorithms,* pp. 93–100.

Schaffer, J. D., and L. Eshelman (1995). Combinatorial Optimization by Genetic Algorithms: The Value of Genotype/Phenotype Distinction. In *Proc. of Applied Decision Technologies* (ADT '95), April 1995, London, pp. 29–40.

Schnier, T., and J. S. Gero (1996). Learning Genetic Representations as an Alternative to Hand-Coded Shape Grammars. In J. Gero and F. Sudweeks (eds.), *Artificial Intelligence in Design '96,* Kluwer, pp. 39–57.

Schoenauer, M. (1996). Shape Representations and Evolution Schemes. In *Proc. of the 5th Annual Conf. on Evolutionary Programming,* MIT Press, pp. 121–129.

Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Stromungstechnik.* Diplomarbeit, Technische Universität Berlin.

Schwefel, H.-P. (1981). *Numerical Optimization of Computer Models.* Wiley.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* Wiley.

Sims, K. (1991). Artificial Evolution for Computer Graphics. *Computer Graphics* 25(4):319–328.

Sims, K. (1994a). Evolving Virtual Creatures. In *Computer Graphics,* Annual Conference Series (SIGGRAPH '94 Proceedings), July 1994, pp. 15–22.

Sims, K. (1994b). Evolving 3D Morphology and Behaviour by Competition. In R. Brooks and P. Maes (eds.), *Artificial Life IV Proceedings,* MIT Press, pp. 28–39.

Sims, K. (1999). Evolving Three-Dimensional Morphology and Behaviour. In P. J. Bentley (ed.), *Evolutionary Design by Computers.* Morgan Kaufmann.

Smith, R. E., and D. E. Goldberg (1992). Diploidy and Dominance in Artificial Genetic Search. *Complex Systems* 6:251–285.

Soddu, C. (1995). Recreating the City's Identity with a Morphogenetic Urban Design. In *17th International Conference on Making Cities Livable,* Freiburb-im-Breisgau, Germany, Sept. 5–9, 1995.

Spector, L., W. B. Langdon, U.-M. O'Reilly, and P. J. Angeline (eds.) (1999). *Advances in Genetic Programming 3.* MIT Press.

Srinivas, N., & K. Deb (1995). Multiobjective Optimization Using Nondominated Sorting in Genetic Algorithms. *Evolutionary Computation* 2(3):221–248.

Stiny, G. (1980). Introduction to Shape and Shape Grammars. *Environment and Planning B* 7:343–351.

Stiny, G., and W. Mitchell (1978). The Palladian Grammar. *Environment and Planning B* 5:5–18.

Syswerda, G. (1989). Uniform Crossover in Genetic Algorithms. In D. Schaffer (ed.), *Proc. of the Third Int. Conf. on Genetic Algorithms,* Morgan Kaufmann.

Taura, T., and I. Nagasaka (1999). Adaptive Growth Type Representation for 3D Configuration Design. In P. J. Bentley (ed.), *First Special Issue on Evolutionary Design, Artificial Intelligence for Engineering Design, Analysis and Manufacturing* (AIEDAM) 13(3):171–184.

Thompson, A. (1995). Evolving Fault Tolerant Systems. In *Genetic Algorithms in Engineering Systems: Innovations and Applications,* IEE Conf. Pub. No. 414, pp. 524–529.

Thompson, A., and P. Layzell (1999). Analysis of Unconventional Evolved Electronics. *Communications of the ACM* 42(4):71–79.

Todd, S., and W. Latham (1992). *Evolutionary Art and Computers.* Academic Press.

Todd, S., and W. Latham (1999). The Mutation and Growth of Art by Computers. In P. J. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann.

Tuson, A. L., and P. M. Ross (1998). Adapting Operator Settings in Genetic Algorithms. *Evolutionary Computation* 6(2).

Vavak, F., and T. Fogarty (1996). Comparison of Steady State and Generational GAs for Use in Nonstationary Environments. In *Proceedings of the IEEE 3rd International Conference on Evolutionary Computation ICEC'96.*

Whitley, D., and T. Starkweather (1990). GENITOR II: A Distributed Genetic Algorithm. *Journal of Experimental and Theoretic Artificial Intelligence* 2(3):189–214.

Yao, X., G. Lin, and Y. Liu (1997). An Analysis of Evolutionary Algorithms Based on Neighborhood and Step Sizes. In P. Angeline, R. Reynolds, J. McDonnell, and R. Eberhart (eds.), *Proceedings of the Sixth International Conference on Evolutionary Programming,* Springer, pp. 298–307.

Yao, X., and Y. Liu (1996). Fast Evolutionary Programming. In L. Fogel, P. Angeline, and T. Bäck (eds.), *Proceedings of the Fifth International Conference on Evolutionary Programming,* MIT Press.

Yu, T., and P. Bentley (1998). Methods to Evolve Legal Phenotypes. In *Fifth Int. Conf. on Parallel Problem Solving from Nature,* Amsterdam, Sept. 27–30, 1998.

Yu, T., and C. Clack (1998a). Recursion, Lambda Abstractions and Genetic Programming. In *Proceedings of Genetic Programming 1998,* Morgan Kaufmann.

Yu, T., and C. Clack (1998b). PolyGP: A Polymorphic Genetic Programming System in Haskell. In *Proceedings of Genetic Programming 1998,* Morgan Kaufmann.

Zitzler, E., and L. Thiele (1999). Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation* 2(4):257–272.

This Page Intentionally Left Blank

I

PART

EVOLUTIONARY CREATIVITY

1 **Creativity in Evolution: Individuals, Interactions, and Environments**
Tim Taylor

2 **Recognizability of the Idea: The Evolutionary Process of Argenìa**
Celestino Soddu

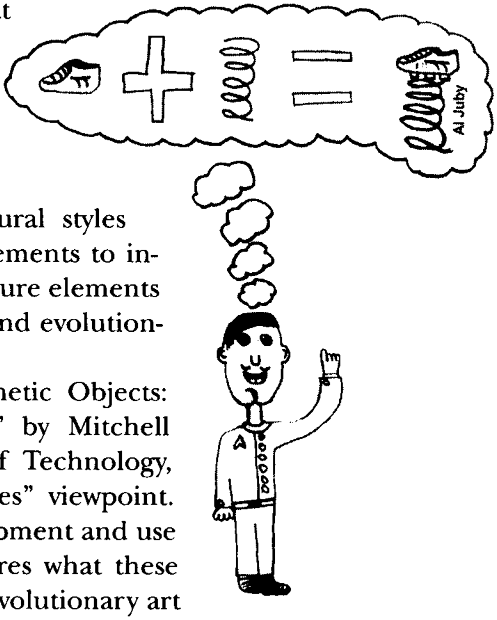
3 **Breeding Aesthetic Objects: Art and Artificial Evolution**
Mitchell Whitelaw

4 **The Beer Can Theory of Creativity**
Liane Gabora

As we saw at the end of the introduction, creativity is a very difficult concept to define. Clearly the 10- and 11-year-old children from North County Primary School who drew the images for these part opener pages are very creative. But can we expect evolution to be this creative—and what would that mean, anyway? Perhaps Al's picture on the next page is right: maybe creativity is the ability to combine two very disparate ideas into a novel solution.

This part explores these thorny issues of evolution and creativity from four contrasting viewpoints. Our first chapter, "Creativity in Evolution: Individuals, Interactions, and Environments," by Tim Taylor of the Institute of Perception, Action and Behaviour, University of Edinburgh, provides the opinion of a computer scientist. Here, Tim considers evolutionary systems that possess an inherent ability to be creative, rather than those in which creativity is achieved by interactions with a human observer. He suggests that open-ended evolution may be crucial to enable the same kinds of creative solution, in our computers that we see in natural systems.

Next, Celestino Soddu, the director of the Generative Design Lab at Milan Polytechnic explains—in his own unique style—his very personal view as a designer, in "Recognizability of the Idea: The Evolutionary Process of Argenìa." Celestino explains how Argenìa is so successful at generating architecture, designs, and art that embody recognizable ideas of people, and why this is important. He describes how his approaches develop architectural styles over time by using existing elements to influence the development of future elements in a dynamic, unpredictable, and evolutionary way.



Chapter 3, "Breeding Aesthetic Objects: Art and Artificial Evolution," by Mitchell Whitelaw of the University of Technology, Sydney, provides a "humanities" viewpoint. Mitchell focuses on the development and use of evolutionary art and explores what these approaches mean to art. Will evolutionary art extend the capabilities of artists, or will it only constrain their imagination?

Finally, Chapter 4, "The Beer Can Theory of Creativity," by Liane Gabora of The Center "Leo Apostel" at the Vrije Universiteit, Brussels, gives a "cultural view." Liane explores how cultural information may evolve and investigates the meaning of concepts such as novelty and creativity. Perhaps our own creativity also has a lot to do with evolution.

1

CHAPTER

Creativity in Evolution: Individuals, Interactions, and Environments

Tim Taylor University of Edinburgh

1.1 INTRODUCTION

What are the basic design considerations for creating an artificial evolutionary system that displays the sort of creativity observed in biological evolution? In this chapter I address such questions, considering evolutionary systems that possess an inherent ability to be creative, rather than those in which creativity is achieved by interactions with a human observer. I start by discussing what I mean by creativity in this context and how it relates to open-ended evolution. I then discuss various issues concerning the design of artificial evolutionary systems and their capacity for creative evolution. The discussion emphasizes that it is necessary to consider not just the design of individuals, but also the sort of environments in which they live, and how individuals can interact with each other and with the physical (i.e., abiotic) environment. Much of this discussion is presented in relation to a hypothetical structure (which I refer to as "proto-DNA") that would be suitable for acting as a robust initial seed for an open-ended, creative evolutionary process. I go on to discuss how these issues should be integrated into a unifying framework in which the study of creative artificial evolutionary systems can be developed.

1.2 CREATIVITY AND OPENED-ENDED EVOLUTION

Most forms of artificial evolutionary system are designed to be used as optimization tools; the course of evolution is guided by an extrinsically defined fitness function that preferentially selects individuals that are deemed to be "fit" according to some specific criterion. Examples include genetic algorithms (Holland 1975), genetic programming (Koza 1992), and similar techniques. In this type of system, the evolving individuals move toward a predefined, and usually static, fitness peak, and when this peak has been reached, they generally stay there.

In contrast, some other evolutionary systems have a less determinate feel. These include models of *co*-evolutionary processes of one form or another, where the success of organisms in one population depends upon the success of organisms in another, coevolving population. Examples of this type of work include systems described by Hillis (1990), Sims (1994), Miller and Cliff (1994), and Floreano, Nolfin, and Mondada (1998). Hillis, for example, coevolved a population of algorithms for sorting lists of numbers, together with a population of lists that were used to test the algorithms. The idea was to reward the algorithms for correctly sorting the test lists and reward the test lists for baffling the algorithms. Therefore, as the algorithms evolved to better deal with the test cases, so the test cases evolved to present harder challenges to the algorithms. One population spurred on the other to higher fitness, and the algorithms obtained by this method were indeed consistently better and faster than those obtained using a fixed set of test cases.

However, these coevolutionary studies are geared toward producing organisms that are good at performing a particular task. To this end, the coevolving organisms are still generally competing in some prespecified (extrinsically defined) game, and they are not given the potential for developing entirely *new* games to play.

Another group of models has moved even further from the idea of extrinsically defined fitness functions, dispensing altogether with the notion of modeling evolution toward any sort of high-level goal. Examples include models by Barricelli (1957), Conrad and Pattee (1970), Packard (1988), Ray (1991), Adami and Brown (1994), and Holland (1995). In these systems, individuals are competing for one or more limited resources that they require in order to survive and propagate (e.g., memory or CPU time). The fact that these resources are limited induces natural (intrinsic) selection for those individuals that outcompete their neighbors. These systems have more of an open-ended nature because the individuals are not evolving toward any predefined high-level goal; they are being selected for their ability to win the limited resources, but this ability is measured *relative to (some or all of) the other individuals in the population.* Hence, an individual's "fitness" changes as new individuals are born and existing ones die. As the biotic environment of an individual (i.e., the other individuals in the population) changes, that individual (or its offspring) must adapt in order to survive. This adaptation, in turn, causes the environment experienced by other organisms to change, so the population is in a constant state of flux. This scenario is equivalent to Van Valen's Red Queen hypothesis for indefinite evolutionary change in biological ecosystems (Van Valen 1973).

For promoting open-ended evolution, the importance of individuals being part of the environment experienced by other individuals has also been emphasized by some members of the artificial life community (for example, Ray 1991; Arthur 1994; Bedau 1998). However, the theoretical considerations driving the design of the above systems have focused almost exclusively on properties of individuals (e.g., the self-reproduction process). Little is said, from a theoretical point of view, of how the environment should be constructed (including how individuals form part of the environment for other individuals), or how individuals should be allowed to interact.

Some of these latter systems can be regarded as modeling "open-ended evolution," in the sense that new, adaptively successful individuals continuously appear in the populations—evolutionary activity does not peter out.[1] However, the *kinds* of evolutionary innovation observed in these systems are generally fairly restricted. For example, the evolutionary innovations observed in experiments with Tom Ray's Tierra platform (described in more detail in Section 1.3.2) fall into two broad categories: "ecological solutions" and "optimizations" (Ray 1997), but the limited interactions between individuals in Tierra restricts the range of possible innovations even within these categories. Much has been said of the evolution of parasites[2] and related ecological phenomena in Tierra (for example, Ray 1991), but the fact that they appear is due to some fairly specific aspects of the system's design and of the particular way in which the ancestral self-reproducing programs were written; these phenomena emerged only because it was very easy for evolution to discover them (see, for example, Taylor 1999). In short, it is hard to escape the feeling that most of these systems are only capable of producing innovations of the "more of the same" variety (e.g., more optimized code), rather than anything fundamentally new.

It is hard to be precise about what counts as "fundamentally new," but I am referring to the ability of individuals to interact with their (biotic and abiotic) environment with few restrictions and to evolve mechanisms for sensing new aspects of this environment and for interacting with it in new ways. This includes the ability of individuals to utilize new physical modalities (e.g., sound, light, electrical conductance) that they previously did not use, to develop new functional relationships with their environment (e.g., the ability to fly), and also for the very notion of individuality to change in radical ways (e.g., the evolution of multicellular organisms from unicellular ones). It is these sorts of evolutionary innovations that I am labeling "creative." Creativity is therefore distinct from open-endedness; a system capable of open-ended evolution is not necessarily creative. Biological evolution has managed all of these feats, so the question is how to instill similar capacities into artificial evolutionary systems.

---

[1] Although even in these systems it is debatable whether this can continue indefinitely.

[2] That is, short programs that are unable to reproduce by themselves, but do so by reading code from neighboring programs.

In the following sections, I analyze the design of artificial evolutionary systems (specifically, those with intrinsic selection) with respect to open-ended evolution. I also consider how the capacity for *creative* evolution can be secured. The analysis emphasizes the need for the explicit consideration of environments and of interactions as well as of individuals.

1.3 DESIGN ISSUES

This section will begin by introducing von Neumann's work on the logic of self-reproduction. Next I shall discuss Ray's Tierra model in more detail. I then analyze self-reproduction in a number of artificial evolutionary systems in terms of von Neumann's proposed architecture. Finally, in Sections 1.3.4 to 1.3.6, I discuss issues relating to phenotypic properties, and the relationship between individuals and the environment in artificial systems.

1.3.1 Von Neumann's Architecture for Self-Reproduction

In the late 1940s and early 1950s, John von Neumann devoted considerable time to the question of how complicated machines could evolve from simple machines.[3] Specifically, he wished to develop a formal description of a system that could support self-reproducing machines that were robust in the sense that they could withstand some types of mutation and pass these mutations on to their offspring. Such machines could therefore participate in a process of evolution.

Inspired by Turing's earlier work (Turing 1936) on universal computing machines, von Neumann devised an architecture that could fulfil these requirements. The machine he envisaged was composed of three subcomponents (von Neumann 1966):

---

[3] Von Neumann had difficulties in defining precisely what the term "complicated" meant. He said, "I am not thinking about how involved the object is, but how involved its purposive operations are. In this sense, an object is of the highest degree of complexity if it can do very difficult and involved things" (von Neumann 1966).

1. A general *constructive* machine, **A**, which could read a description $\phi(\mathbf{X})$ of another machine, **X**, and build an instance of **X** from this description:

$$\mathbf{A} + \phi(\mathbf{X}) \Rightarrow \mathbf{X} \tag{1}$$

   where + indicates a single machine composed of the components to the left and right suitably arranged, and ⇒ indicates a process of construction.

2. A general *copying* automaton, **B**, which could copy any instruction tape:

$$\mathbf{B} + \phi(\mathbf{X}) \Rightarrow \phi(\mathbf{X}) \tag{2}$$

3. A *control* automaton, **C**, which, when combined with **A** and **B**, would first activate **B**, then **A**, then link **X** to $\phi(\mathbf{X})$ and cut them loose from $(\mathbf{A} + \mathbf{B} + \mathbf{C})$:

$$\mathbf{A} + \mathbf{B} + \mathbf{C} + \phi(\mathbf{X}) \Rightarrow \mathbf{X} + \phi(\mathbf{X}) \tag{3}$$

   Now, if we choose **X** to be $(\mathbf{A} + \mathbf{B} + \mathbf{C})$, then the end result is

$$\mathbf{A} + \mathbf{B} + \mathbf{C} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C}) \Rightarrow \mathbf{A} + \mathbf{B} + \mathbf{C} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C}) \tag{4}$$

This complete machine plus tape, $[\mathbf{A} + \mathbf{B} + \mathbf{C} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C})]$, is therefore self-reproducing. From the point of view of the evolvability of this architecture, the crucial feature is that we can add the description of an arbitrary additional automaton **D** to the input tape. This gives us

$$\begin{gathered}\mathbf{A} + \mathbf{B} + \mathbf{C} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}) \Rightarrow \\ \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})\end{gathered} \tag{5}$$

Furthermore, notice that if the input tape $\phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$ is mutated in such a way that the description of automaton **D** is changed, but that of **A**, **B**, and **C** are unaffected—that is, the mutated tape is $\phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}')$—then the result of the construction will be

$$\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D} + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}) \overset{\textit{mutation}}{\Rightarrow} \mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}' + \phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}') \tag{6}$$

The reproductive capability of the architecture is therefore robust to some mutations (specifically, those mutations that only affect the description of **D**), so the machines are able to evolve. Von Neumann pointed out that the action of the general copying automaton, **B,** was the decisive step that gave his architecture the capacity for evolving machines of increased complexity because **B** is able to copy the description of any machine, no matter how complicated (von Neumann 1966, p. 121). This ability is clearly demonstrated in Reaction (5) above.

The original implementation envisaged by von Neumann was a constructive system, which Burks has referred to both as the "robot model" and as the "kinematic model" (Aspray and Burks 1987, p. 374). However, von Neumann decided that the system was too complicated to capture in a set of rules that were both simple and enlightening, so he turned his attention to developing the cellular automata (CA) framework with Stanislaw Ulam. Von Neumann described the detailed design of a self-reproducing machine in a cellular automata space, according to the architecture described above.[4] In this CA model, each of the basic components of von Neumann's architecture, **A, B, C,** and ϕ, were represented as particular configurations of cell states within a two-dimensional lattice of cells, and the action of the cells was defined (as in all CA models) by the particular transition functions used to determine how a cell's state changed over time. Recently, a slightly modified and simplified version of this design was implemented on a computer (Pesavento 1995). One of the major achievements of von Neumann's work was to clarify the logical relationship between *description* (the instruction tape, or genotype) and *construction* (the execution of the instructions to eventually build a new individual, or phenotype) in self-replicating systems. However, as already mentioned and as emphasized recently by McMullin (1992), his work was always within the context of self-replicating systems that would also possess great *evolutionary* potential.

1.3.2 Tierra

An artificial evolutionary system of a somewhat different design than von Neumann's that has received a great deal of attention in the last decade is Tom Ray's Tierra model (Ray 1991), mentioned in Section 1.2. Tierra is an implementation of a virtual parallel computer that can simulate the concurrent execution of many hundreds of programs. The programs are written in a specially designed language that is both robust and simple. Programs written in this language can be mutated (i.e., random changes can be made to them) without causing the computer to crash.

---

[4] The general constructive machine **A** of this design is often referred to as a "universal constructor." However, this term should be used with caution; from the above description of the architecture, it is clear that **A** can build any machine **X** *that can be described upon a tape* $\phi(X)$. For cellular automaton models it can be proved that there are some configurations that the universal constructor cannot build (for example, Moore 1962; Myhill 1963). These are referred to as "Garden of Eden" configurations, as the only way they may exist is if they are programmed in as the initial state of the space at time zero.

An evolutionary run commences with the introduction of an *ancestor* program into the otherwise empty memory. The ancestor is a handwritten self-replicator that produces another copy of itself in the computer's memory when it is run. At each iteration of the system, each program in the computer's memory is allowed to execute a certain number of instructions. A small element of stochastic behavior is associated with the execution of the machine instructions; for example, an `add` instruction that usually adds one to its operand may occasionally add zero or two instead, or a `copy` instruction may sometimes mutate a byte of the data it is copying. The programs are also subject to point random mutations at a given low rate. In either of these ways, as a run proceeds, variations of the ancestor program begin to appear. If a variation retains the ability to produce a copy of itself, then it too may be retained in the population of programs over a number of generations. As the available memory begins to fill, a "reaper" operation is performed to kill off a number of the programs. Programs that perform operations that cause their flag to be set[5] are killed off more quickly than others (by being advanced up the "reaper queue"), but otherwise the order in which programs are killed off is largely determined by their age.

As mentioned in Section 1.2, a number of interesting results have been obtained from such evolutionary runs. For example, "parasites" have appeared—short pieces of code that run another program's copying procedure in order to copy themselves. Hyperparasites (parasites of parasites) have also been observed, along with other interesting ecological phenomena (Ray 1991).

Although Tierra was designed to study evolution, and in particular (originally, at least) the evolution of multicellular organisms from unicellular ones, it was not built around any particular theory of what the important features of this transition might have been. There are therefore no coherent theoretical reasons for deciding which features should be modeled and which should be left out. This weakness is not specific to Tierra, but is shared by most, if not all, of the other Tierra-like systems that have emerged over the last decade (for example, Taylor 1997).

In describing the philosophy behind the Tierra system, Ray explains that

---

[5] Examples include issuing a `jmp` instruction with a template pattern for which no match can be found and attempting to write to a memory address for which the program does not have write access.

> . . . rather than attempting to create prebiotic conditions from which life may emerge, this approach involves engineering over the early history of life to design complex evolvable organisms, and then attempting to create conditions that will set off a spontaneous evolutionary process of increasing diversity and complexity of organisms (Ray 1991, p. 373).

However, in order to "engineer over" several billion years of evolution, we would need to have a very good idea of the design and behavior of the resulting organisms, and an understanding of why they had evolved in such a way (in order to know which aspects of their design and behavior were the most important for us to model).[6] Unfortunately we do not possess such details of the organisms that existed at this stage of evolution on Earth.

I am certainly not the first person to criticize artificial life models on these grounds. For example, Howard Pattee warns that "simulations that are dependent on ad hoc and special-purpose rules and constraints for their mimicry cannot be used to support theories of life" (Pattee 1988, p. 68).

To be fair, Ray does offer a definition of life in his work with Tierra. He says, "I would consider a system to be living if it is self-replicating, and capable of open-ended evolution" (Ray 1991, p. 372). However, determining necessary and sufficient conditions for a system to be capable of open-ended evolution is half of the problem, and Ray's definition tells us nothing about how we should go about building such a system. This being the case, the definition does not provide an adequate theoretical grounding for Tierra and similar models.

A weakness of Ray's definition of life for our present purposes is that it does not define what sorts of environments might support life, or the sorts of ecological interactions that should be available. It is often argued that ecological processes may play a primary role in promoting evolutionary activity and the evolutionary increase of complexity of some organisms (for an overview, see Taylor 1999, Section 2.3.1). Furthermore, some of the most spectacular examples of artificial evolution rely upon coevolutionary interactions between organisms, as mentioned in Section 1.2. This suggests that we should think more carefully about such issues, rather than treating them in the rather ad hoc way that has often been used in the past. This point has been made by Pattee, who says:

> . . . life must have arisen and evolved in a nonliving milieu. In real life we call this the real physical world. If artificial life exists in a computer, the computer milieu must define an artificial physics . . . What is an artificial physics or physics-as-it-could-be? Without principled restrictions this question

---

[6] Ray himself recognizes these difficulties, but is more optimistic that they can be overcome (Ray 1991, p. 399).

> will not inform philosophy or physics, and will only lead to disputes over nothing more than matters of taste in computational architectures and science fiction (Pattee 1995a, p. 29).

The ad hoc feel of Tierra-like systems is a direct consequence of this lack of theoretical grounding. The unmanageable parameter space of many of them can also be attributed to this lack of direction. As a result of these weaknesses, even if interesting behaviors are observed in these systems, we are unlikely to be able to adequately explain them in any general sense without further substantial theorizing and experimentation. It may be that a model of self-replication and open-ended evolution is necessarily somewhat complex, but, even if this is so, the theoretical framework upon which it is built should prescribe the implementational details as much as is practically possible.

1.3.3 Implicit versus Explicit Encoding

Tierra, as well as many of the other artificial evolutionary systems mentioned in Section 1.2, can be analyzed in terms of von Neumann's work. In this section I analyze Tierra in terms of the various components (e.g., **A**, ϕ) of his architecture. Specifically, I consider the extent to which these components are explicitly encoded on the evolving individuals themselves, rather than being implicitly encoded in the "laws of physics" of the environment in which they exist (i.e., the operating system of the platform).

Now, as we are interested in the evolution of the self-reproducing individuals in these systems, and as the inheritable information of each individual (i.e., the part that gets passed on from parent to offspring) is contained on the tape ϕ in von Neumann's architecture, I will assume that the tape must be explicitly represented in some fashion, otherwise there would be nothing that could evolve. We can now ask *which parts* of the $[\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D}]$ architecture are explicitly encoded on the tape ϕ. Of course, even the behavior of those parts that are represented on the tape will still to some extent be encoded in the "laws of physics" of the environment, but I think the analysis is nevertheless worthwhile.

In the case of von Neumann's envisaged implementation of self-reproducing cellular automata, it is clear that all four subcomponents (i.e., **A**, **B**, **C**, and **D**) are very explicitly encoded on the tape $\phi(\mathbf{A} + \mathbf{B} + \mathbf{C} + \mathbf{D})$; the environment in which the automaton exists implicitly encodes only very low-level actions in the form of the local transition rules of individual cells.

I would suggest that the reproducing programs in Tierra and similar systems can also sensibly be analyzed in terms of von Neumann's architecture.

Before I begin, I would like to make a couple of general points, which might help to reorient the reader to my perspective. First, I believe that the notion of a phenotype fundamentally involves *interaction* with the environment (and that this is the essential distinction between the notions of phenotype and genotype—the latter being an informational concept). When I talk about phenotypes in the following, therefore, and specifically when I talk about the automata **A, B, C,** and **D,** I am interested in the role these phenotypic structures play—their function—rather than the details of implementation or of how that function is achieved. Second, note that the terminology commonly used to describe reproducers in Tierra-like systems is somewhat different than that used for von Neumann's work. Because of the similarity between Tierra-like operating systems and those of standard digital computers, the actions of Tierran reproducers are often referred to as "computations" rather than "constructions," even when a reproducer is in the process of building a new copy of itself. However, this process of reproduction is, of course, central to the Tierra approach, and I believe that this procedure of building a copy of a program in a different part of memory is, in all the relevant details, a process of construction in just the same way as construction processes in von Neumann's cellular automata model. In the following, also remember that von Neumann's general constructing automaton **A** is the machinery that *interprets* the tape to produce a new machine (phenotype), and the general copying automaton **B** copies the tape uninterpreted.

At first sight it might seem that there is no separate genetic description of the program in a Tierra-like system. The picture is complicated by the fact that the machinery that interprets the program (i.e., automaton **A**) does not reside in the same part of the computer in which the program itself is stored. The state information for this machinery—a program's "virtual CPU" (i.e., the instruction pointer, stacks, registers, etc.)—is generally represented in an independent area of memory from the program's instructions. Furthermore, the actual "interpreting machinery" of the virtual CPU is encoded in the global operating system provided by the platform and is in this sense implicit in the program's environment. Additionally, the control automaton **C,** which controls when the instructions in the program get executed, is also implicit in the part of the operating system that governs mechanisms such as how a program's instruction pointer is updated after the execution of each instruction. All that is left to be explicitly encoded by the program, therefore, is the copying automaton **B,** and potentially any other arbitrary automaton **D.**

The instructions that make up the program exist in an unreactive state in the system's random-access memory. It is only when the control automaton **C** transfers instructions to the interpreting automaton **A** that they become "active."

Looked at in this way, we can see that it is the *behavior* of the program (including looping, jumping, etc.) that is the result of automaton **A** interpreting the unreactive genetic description. This behavior is therefore the equivalent of the constructed machine (and the actions it performs—i.e., the phenotype) in von Neumann's design, and the string of instructions residing in the random-access memory (which is normally referred to as the program) is the tape or genetic description of this phenotype. It is perhaps easier to see the distinction if one considers a parallel program, with multiple processes (with different state information) using the same program listing.

I therefore suggest that a self-reproducing program in a Tierra-like system is consistent with von Neumann's architecture. However, as automata **A** and **C** are largely implicit in the environment in which the programs reside (the only explicit representation being the state information in a program's virtual CPU) and are certainly not encoded by the individual programs, we can see that a "program," in the sense of a string of instructions in the system's random-access memory, corresponds to the tape $\phi(\mathbf{B} + \mathbf{D})$ in von Neumann's scheme.

It is interesting to speculate on what information we might desire to be explicitly encoded on a structure that would be suitable for acting as a robust initial seed for an open-ended, and possibly creative, evolutionary process. I will refer to such a structure as "proto-DNA." We would like our proto-DNA to be an indefinite hereditary replicator if it is to be such a seed (Maynard Smith and Szathmáry 1995). In other words, it should be able to exist in an unlimited number of configurations that retain the ability to reproduce. If the copying process is encoded on the tape itself, then mutations have the potential to disrupt its ability to be reproduced. It would therefore seem desirable that the copying automaton **B** of our proto-DNA be largely implicitly encoded in the environment. Note that this would not necessarily prevent a more complicated, and possibly more reliable, explicit copying process $\mathbf{B}'$ later evolving from (but still based upon) the simpler implicit process, as indeed seems to have happened during biological evolution.

If the copying procedure for our proto-DNA is implicitly encoded in the environment, however, any configuration of proto-DNA would, all else being equal, be able to reproduce as well as any other. In other words, there would be no basis for preferentially selecting some configurations over others, and therefore no basis for an evolutionary process. Specific configurations of proto-DNA must therefore have some specific properties that are selectively significant. Models of the origin of life commonly presume that these simple phenotypic properties were things such as increased stability of the molecule, simple control of the local environment, catalytic activity, and so on (for example, Eigen and Schuster 1977; Cairns-Smith 1985; Szathmáry and Demeter 1987).

At the initial stages of an evolutionary process, however, we would not expect there to be mechanisms for explicitly decoding the proto-DNA; in other words, the interpretation machinery **A** is implicit.[7] This means that particular configurations of proto-DNA should have some specific phenotypic properties (such as the ability to act as catalysts) that can be determined directly from their structure rather than having to be explicitly decoded from the genotype. We could therefore regard the proto-DNA as merely $\phi(\mathbf{D})$, meaning that particular configurations have particular phenotypes associated with them, which (1) are not related to the process of self-reproduction per se and (2) do not require to be decoded by an explicit interpretation automaton **A**. Regarding the kinds of simple phenotypes that we might wish to be available to our proto-DNA, some possibilities are suggested by the origin-of-life models mentioned previously, but in general the options seem endless. Graham Cairns-Smith observes:

> It is almost too easy to imagine possible uses for phenotype structures—because the specification for an effective phenotype is so sloppy. A phenotype has to make life easier or less dangerous for the genes that (in part) brought it into existence. There are no rules laid down as to how this should be done (Cairns-Smith 1985, p. 106).

Proto-DNA with inherent phenotypic properties can therefore serve as a suitable starting point for an open-ended evolutionary process. To digress a little, with regard to the issue of how symbolic information arises in evolution (discussed, for example, by Pattee 1995b), this requirement ensures that the matter-symbol relationship is inherent in the system from the beginning. The material is selected for its phenotypic properties, but it is its genetic information that is passed on to its offspring. In this situation, it is necessary to assume that by inheriting this genotype, the offspring will also share the phenotypic properties. For example, in a simple RNA-world scenario,[8] we could imagine that molecules that inherit a particular sequence of bases would adopt a particular three-dimensional structure, which might, say, confer specific catalytic properties (as demonstrated by Zaug and Cech 1986). We could therefore regard the genetic information (the sequence of bases on the RNA molecule) as a symbolic representation of its phenotypic properties (its catalytic action in this example).

However, there would presumably be a limit on the number of different inherent phenotypic properties these proto-DNA structures might possess. Furthermore, if the proto-DNA is to reliably reproduce, it should be a fairly stable molecule, and this requirement further restricts the range of effective phenotypic properties that it might have. If more complicated phenotypes are to arise later on in the evolutionary process, therefore, it appears necessary that a stronger distinction is introduced between genotypes and phenotypes. The biologist C. H. Waddington remarked that:

---

[7] We could, of course, "hard-wire" explicit interpretation machinery into the system (as in the programming language provided in Tierra), but to do so would inevitably impose restrictions on the evolutionary possibilities available.

[8] For references to work on RNA worlds, see Nuño et al. (1995) and Lazcano (1995).

> . . . in practice—and perhaps because of a profound law of action-reaction—it is difficult (impossible?) to find a [molecule] which is stable enough to be an efficient store and at the same time reactive enough to be an efficient operator (Waddington 1969, p. 115).

The advantages of a genotype-phenotype distinction over other forms of reproduction have been discussed by many people (for a review, see Taylor 1999, Section 7.2.3). For such a distinction to arise with proto-DNA, we require that it at least has the potential for explicit interpretation machinery **A**′ and control machinery **C**′ to become associated with it. This would involve some form of specific reaction to subsections of information in the proto-DNA, but more work is needed to fully identify how this potential for explicit interpretation might be assured.

1.3.4 Ability to Perform Other Tasks

In the previous section it was suggested that proto-DNA in its primitive form should not involve much interpretation or control machinery. However, it is important that some specific phenotypic properties are implicitly associated with specific structures (i.e., these properties are apparent without the need for explicit interpretation machinery). Without the ability of individual replicators to have other properties as well as self-reproduction, the evolving system will not be very interesting. Indeed Muller, who in the early part of this century was the first person to explicitly propose an exclusively evolutionary definition of life, emphasized the importance of this material "affecting other materials and, therewith, its own success in genetic survival" (Muller 1966, p. 512). This picture of individual reproducers affecting other materials reminds us that biological evolution has involved the coevolution of interacting organizations rather than of single, isolated reproducers. As mentioned in Section 1.3.2, many existing artificial evolutionary systems have concentrated almost exclusively on modeling individuals, with little regard for the principled modeling of interactions between individuals.

Nils Barricelli was well aware of the need for reproducers to perform other tasks when he designed his artificial life platform in the early 1950s. He wrote:

> It may appear that the properties one would have to assign to a population of self-reproducing elements in order to obtain Darwinian evolution are of a spectacular simplicity. The elements would only have to: (1) Be self-reproducing and (2) Undergo hereditary changes (mutations) in order to permit evolution by a process based on the survival of the fittest (Barricelli 1962, pp. 70–71).

He went on to describe a simple discrete one-dimensional model where each cell is either empty or contains an integer number. The numbers reproduce according to the implicit rules of the system, and mutations arise under certain circumstances. This simple model therefore fulfils the fundamental requirements for an evolutionary process. However, as Barricelli noted, this model of evolution

> . . . clearly shows that something more is needed to understand the formation of organs and properties with a complexity comparable to those of living organisms. No matter how many mutations occur, the numbers . . . will never become anything more complex than plain numbers (*ibid.* p. 73).

Barricelli therefore concentrated on looking for the "missing ingredient."[9] It should be noted that von Neumann, also, was not so much interested in machines that could only self-reproduce, but rather in machines that could perform other tasks as well (von Neumann 1966, p. 92; McMullin 1992, pp. 174–175).

The preceding arguments are leading us in the direction of requiring a form of proto-DNA that reproduces due to the implicit laws of the environment in which it exists, but that also explicitly specifies some properties that can be selected for or against in an evolutionary process. At this point we might note that artificial evolutionary systems that have just these properties already exist, and indeed their use is widespread; these are the optimization tools mentioned in Section 1.2, such as genetic algorithms (for example Holland 1975; Goldberg 1989), genetic programming (Koza 1992), and similar techniques. The difference is that we require a system with the potential for a large degree of *intrinsic* adaptation for open-ended evolution, rather than a system where the selection of individuals is determined by an externally defined fitness function. Intrinsic adaptation is introduced when the *domain of interaction* of the individuals is within the evolving system itself, and the individuals are competing for limited resources. This is in contrast to systems with an explicitly defined fitness function, where the replicators do not directly interact with other replicators. Ray recognized this point himself when discussing the design of artificial life platforms:

---

[9] His solution was to require that elements could only reproduce in symbiotic association with other elements. While this may indeed be an important aspect of the "missing ingredient," it is extremely doubtful that it is the *only* important aspect.

> What all of this discussion points to is the importance of imbedding evolving synthetic organisms into a context in which they may interact with other evolving organisms. A counter example is the standard implementations of genetic algorithms in which the evolving entities interact only with the fitness function, and never "see" the other entities in the population. Many interesting behavioral, ecological and evolutionary phenomena can only emerge from interactions among the evolving entities (Ray 1994, Section 11.1).

Similar arguments for proto-DNA with the properties of implicit reproduction and the potential for explicitly encoded attributes with selective significance have been put forward by Barry McMullin, who points out the connection with Cairns-Smith's general model (Cairns-Smith 1985) for the origin of terrestrial life based upon inorganic information carriers (McMullin 1992, p. 267).

1.3.5 Embeddedness in the Arena of Competition and Richness of Interactions

In the preceding sections I have emphasized the importance of the distinction between intrinsic and extrinsic selection. I will now discuss some issues involved in this distinction in more detail.

An essential requirement for an evolutionary process is that some form of selection mechanism exists, so that some variations of the reproducing entities are favored over others. The selection mechanism therefore introduces a form of competition between the individual reproducers; they become engaged in a struggle for existence. The presence of such a mechanism implies that, in some form, the individuals coexist in an arena of limited capacity, and that they are competing with their neighbors (either globally or locally) for the right to be there.

An evolutionary system must therefore have an arena of competition of some description, although there are few restrictions on the particular form it should take. All that is required is that it introduces the concept of a resource (1) that is a vital commodity to individuals in the population; (2) that is of limited availability; and (3) that individuals can compete for (at either a global or local level). This resource can usually be interpreted as energy, space, matter, or a combination of these.

An issue that arises when considering different evolutionary systems is the extent to which individuals are embedded in this arena of competition. In von Neumann's cellular automata design, individuals are fully embedded—there is no "hidden" state information (i.e., information that is not embedded in the cellular space itself). The same can be said of the biosphere, at least according to materialism. At the other extreme, individuals in a genetic algorithm (GA) have minimal embeddedness—the arena of competition merely contains placeholders for the chromosomes, and the restriction is generally on the number of individuals, regardless of their size (although most GAs have constant-size chromosomes anyway). These two extremes, together with intermediate situations arising in Tierra and Avida,[10] are depicted in Figure 1.1. Note that individuals in Avida are not really embedded in the arena of competition at all; the two-dimensional environment only holds pointers to the cells, in much the same way as in a GA.[11] In Tierra, a program's instructions are embedded in the arena, although each program still has some additional state information (its "virtual CPU" state). In Avida, the fundamental space limitation applies to the number of *programs* that can fit in the arena of competition, whereas in Tierra it applies to the total number of *instructions* contained in all of the programs in the population.

It should be emphasized that this notion of embeddedness is unrelated to the distinction between implicit and explicit encoding, which concerns the degree to which a process is governed by the environment as opposed to a specific object situated within that environment. The issue of embeddedness concerns the representation of individuals only; it does not (directly) concern the representation of the abiotic environment.

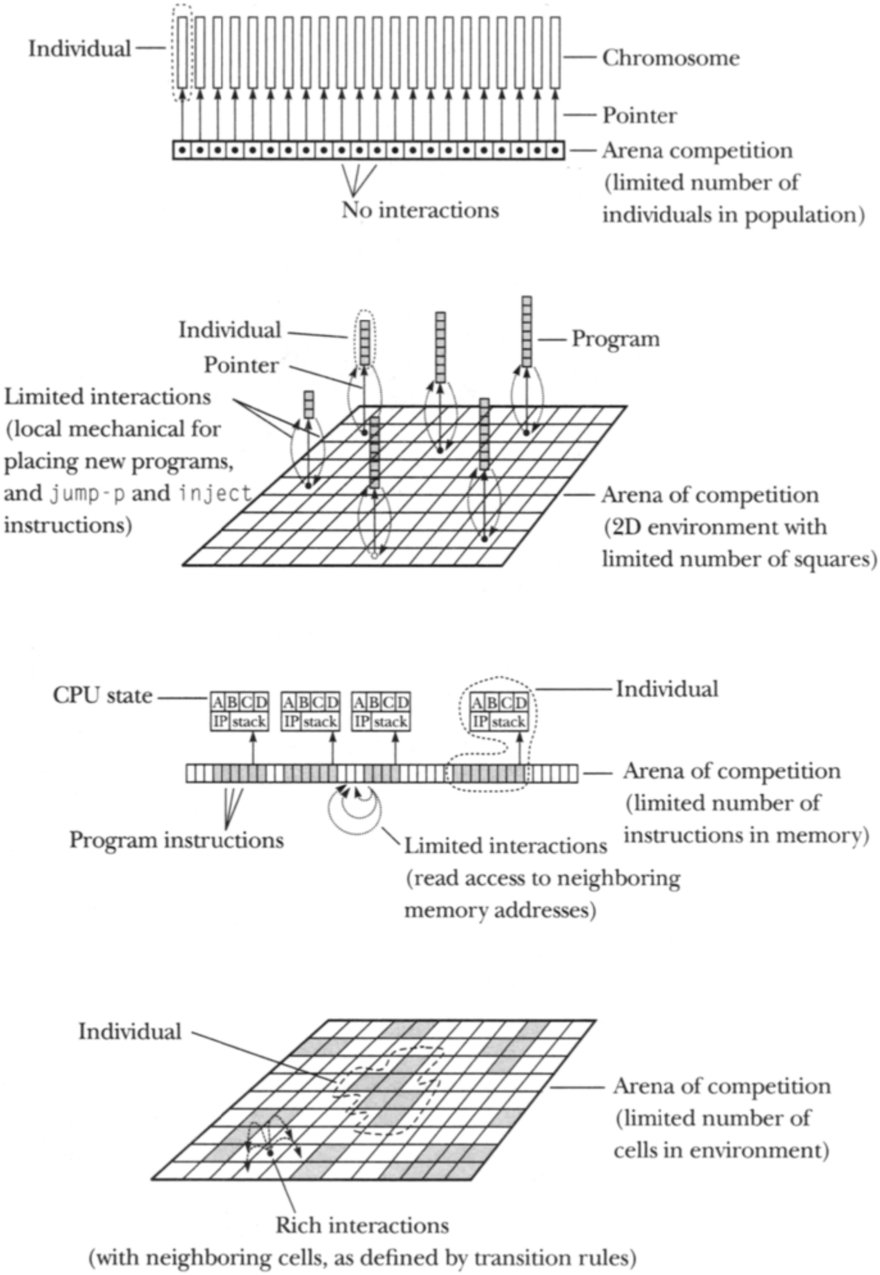
Related to the issue of physical embeddedness is that of how restricted is the range of interactions that are allowed between objects within the arena. In a standard GA, no direct interactions are allowed between chromosomes at all; the continued existence of an individual is decided by the extrinsically defined selection mechanism. As already mentioned, in the default configuration of Avida, programs cannot read the instructions of their neighbors. However, some extra instructions can be enabled to allow these sorts of interactions to occur.[12]

---

[10] Avida is an artificial life platform developed by Chris Adami and colleagues (see *http://www.krl.caltech.edu/avida/*). It is based upon Tierra, but there are some significant differences, especially in the modeling of the environment. For example, individual programs occupy positions in a two-dimensional arena and are only in direct competition for space with their neighbors.

[11] That is, the two-dimensional environment in which all of the programs coexist is distinct from the one-dimensional memory in which each individual program is stored. Furthermore, in the default settings of Avida, programs cannot read instructions of neighboring programs, so no parasitism of this nature can emerge.

[12] These are the `jump-p` and `inject` instructions.


Individual
Chromosome
Pointer
Arena competition
(limited number of
individuals in population)
No interactions
Individual
Program
Pointer
Limited interactions
(local mechanical for
placing new programs,
and jump-p and inject
instructions)
Arena of competition
(2D environment with
limited number of squares)
CPU state
A B C D
IP stack
Individual
Arena of competition
(limited number of
instructions in memory)
Program instructions
Limited interactions
(read access to neighboring
memory addresses)
Individual
Arena of competition
(limited number of
cells in environment)
Rich interactions
(with neighboring cells, as defined by transition rules)

1.1
FIGURE

Embeddedness of individuals and richness of interactions in various artificial evolutionary platforms.

Although programs in Tierra are embedded in the arena of competition to a much greater extent than they are in Avida, the range of interactions allowed with neighboring programs is still fairly restricted; programs can read the code of their neighbors, but they cannot directly write to neighboring memory addresses.

In contrast, von Neumann's cellular automata implementation is far less restrictive; the transition rules of the cellular automata define neighborhood interactions that occur at the level of individual cells and that therefore do not respect boundaries between individual organisms. This is of course similar to the situation of biological organisms, which have the freedom to interact with their environment in a variety of ways only limited by the laws of physics (although organisms themselves generally evolve mechanisms to restrict such free interaction).

From the point of view of the evolvability of individuals, the more embedded they are, and the less restricted the interactions are, then the more potential there is for the very *structure* of the individual to be modified. Recall that this is one aspect of my definition of *creative* evolution. Sections of the individual that are not embedded in the arena of competition are "hard-wired" and likely to remain unchanged unless specific mechanisms are included to allow them to change (and the very fact that specific mechanisms are required suggests that they would still only be able to change in certain restricted ways).

Additionally, from an epistemological point of view, Pattee (1995b) points out that symbolic information (such as that contained in an organism's genes) has "no intrinsic meaning outside the context of an entire symbol system as well as the material organization that constructs (writes) and interprets (reads) the symbol for a specific function, such as classification, control, construction, communication . . ." He argues that a necessary condition for an organism to be capable of creative open-ended evolution is that it encapsulates this entire self-referent organization (Pattee refers to this condition as *semantic closure*). From this it follows that organisms should be constructed "with the parts and the laws of an artificial physical world" (Pattee 1995a, p. 36).[13] In other words, the interpretation (phenotype) of the symbolic information (genotype) of an artificial organism should be constructed and act within the artificial physical environment of the system. Additionally, if the system is to model the *origin* of genetic information, then the genotype itself must also be embedded within the environment; that is, the complete semantically closed organization—the *entire organism*—must be completely embedded within the physical environment. Pattee's arguments also suggest the need for material, rather than purely formal, models—an issue to be discussed in Section 1.3.6.

---

[13] Although he also stresses that "some epistemic principles must restrict physics-as-it-could-be if it is to be any more than computer games" (Pattee 1995a).

To end this section, let us look at Holland's (1995) work with the "Echo" model of complex adaptive systems. Echo possesses many of the features that I have just argued are desirable for a model of open-ended evolution. For example, selection is determined intrinsically by interactions between Echo organisms (or to use Holland's terminology, "agents"), rather than by an externally defined fitness function; the process by which agents reproduce is implicitly defined in the Echo operating system rather than being explicitly encoded by individual agents; and the agents are able to perform a variety of phenotypic behaviors. Echo is also designed upon more explicit design considerations than were most earlier artificial life models; the considerations for Echo are based upon a core set of principles that Holland believes are common to all complex adaptive systems. For all these reasons, I believe Echo represents a significant advance. However, the structure of the individual agents—the notion of what it is to be an agent—is still predefined, and the representation of agents is not fully embedded in the arena of competition. Additionally, the interpretation of agents' chromosomes is handled implicitly by the operating system. Now, the system was designed in this way because it is primarily intended as a general model of complex adaptive systems, rather than a specific model of biological evolution. Indeed, the various successful applications of Echo (for example, Schmitz and Booth 1996; Hraber and Milne 1997) testify to the value of the particular way in which the organism and environment structure have been predefined; if no higher-level structure were imposed, it would be difficult to model most complex adaptive systems of interest (e.g., ecologies, economies, etc.).

In the context of open-ended evolution, however, the design still has some shortcomings. The fact that the Echo operating system implicitly interprets the agents' chromosomes means that they can never come to encode anything more than the fixed range of actions (e.g., offense, defense, conditional exchange of resources) predefined by the designer. In *Hidden Order*, Holland discusses how new meaning can arise in a system, but acknowledges that Echo is deficient in this respect (Holland 1995, p. 138). As Pattee has suggested, it is only when an organism's genotype, phenotype, and the interpretation machinery that produces the latter from the former (including the whole developmental process through which an adult phenotype is produced)—that is, the whole semantically closed organization—is all embedded in the arena of competition that fundamentally new symbolic information can arise in the genome (i.e., the generation of genetic information representing new functional relationships between the organism and its environment). In the discussion of the desirable properties of proto-DNA in Section 1.3.3, it was suggested that this too would initially be interpreted implicitly. It was, however, stressed that the potential should exist for explicit interpretation machinery to evolve, thereby creating an explicit representation of the whole semantically closed organization and allowing the possibility for new symbolic information to arise.

1.3.6 Materiality

The arguments in the previous sections are bringing us to the fundamental question of how matter is represented in these models. If there is a representational distinction between organisms and the environment in which they exist (which comes about by having a hard-wired organism structure and by restricting ecological interactions), some of the fundamental concepts associated with living beings, such as competition for resources, self-maintenance, and so on, become ill-defined. As mentioned in Section 1.3.2, these kinds of ecological relationships may play a very important role in promoting open-ended (and possibly creative) evolution. It is therefore vital that we consider the issues involved in modeling such relationships if we hope to design artificial systems that have the capacity for open-ended and creative evolution.

One of the tenets of Darwinism is that organisms are engaged in a struggle for existence. However, it is difficult to identify the precise nature of this struggle, as Darwin himself observed. In *The Origin of Species,* he wrote:

> What checks the natural tendency of each species to increase in number is most obscure . . . The amount of food for each species of course gives the extreme limit to which each can increase; but very frequently it is not the obtaining food, but the serving as prey to other animals, which determines the average numbers of a species (Darwin 1859, pp. 119–120).

Thus, an important aspect of the struggle for existence is the obtaining of food not from passive, abiotic sources, but through predator-prey relationships. In the biological realm, the struggle for existence involves organisms *killing* other organisms because *the very stuff from which they are constructed is a valuable resource of matter and energy.* This competition is therefore very much a matter of life or death.

It may be difficult to identify the precise nature of the struggle for existence, but it seems likely that the numerous forms of competition can be categorized in terms of a small number of fundamental resources (as mentioned in Section

1.3.5). In the biosphere, a (speculative) list might be matter, energy, space, and information.[14]

Tierra-like systems generally do not have any notion of competition for matter. Indeed, they cannot really be said to have a notion of matter at all, in terms of fundamental units from which all structures are built, and which are conserved during reactions. Instead, when a program is writing a copy of itself, it can produce the copied instructions spontaneously rather than first having to collect a copy of the individual instruction from elsewhere in memory. In other words, the individual instructions are represented as states of specific memory locations, rather than as units of matter, as is also the case in von Neumann's cellular automata model; these systems are formal models rather than material models. The only fundamental competition that exists in Tierra is for space (memory) into which to divide. This is allocated at a global level by the Tierran operating system's memory allocation services.

In Tierra, programs are not even really competing for energy (CPU time) because any number of programs are allowed to execute instructions at each time slice; the limiting factor is how many programs can fit into the available memory. In Avida the situation is somewhat different, as programs can win more CPU time by successfully performing certain arithmetic challenges presented to them by the environment (Adami and Brown 1994).

Programs in Tierra *can* act as resources for other programs in another way, by acting as "library code" that can be read by their neighbors (as happens in the evolution of parasites). In other words, they can act as information resources. However, this is not as strong an ecological interaction as when one organism acts as a resource of matter or energy, in the sense that acting as an information resource is not a direct matter of life or death for the host.

The issue of how energy is represented in these systems is perhaps more controversial. Some would claim that it is essential to model certain fundamental energetic considerations (for example, Morán et al. 1997; Ruiz-Mirazo, Moreno, and Morán 1998). An important point to note is that *all* artificial life platforms have to model energy at the basic level of determining when a component can perform an action (e.g., when a program can execute an instruction, as determined by the system's CPU time allocation scheme). Without a theoretical grounding, any scheme is just as arbitrary as any other (e.g., the schemes in Tierra and Avida). Ideally, the system's design should be based upon explicit considerations of how energy should be modeled.

---

[14] For example, a virus requires information contained in its host's genome in order to reproduce. This information is more than the matter from which the host's DNA is constructed; it involves a particular ordering of matter.

Only when one organism can act as a resource of energy and matter for other organisms do ecological concepts such as food webs and trophic levels (which can act as important drives for open-ended evolution) become relevant.

Other advantages of material evolutionary systems over purely formal systems have been suggested by Luis Rocha:

> Material sign systems are not universal and cannot represent anything whatsoever, but this turns out to be their greatest advantage. The price to pay for the universality of formal symbol systems is complete specificity, that is, full description of its components and behaviour. Conversely, material sign systems are built over certain building blocks which do not need a description. For instance, DNA does not need to encode anything but aminoacid chains, there is no need to include in genetic descriptions information regarding the chemical constituents of aminoacids nor instructions on how to fold an aminoacid chain—folding comes naturally from the dynamical self-organization of aminoacid chains (Rocha 1998).

In other words, the genome does not have to encode information about every aspect of the organism's phenotype because some features will just fall into place "for free," due to the self-organizational properties of the constituent matter. This may significantly ease the problem of evolving complex phenotypes.

1.4 A FULL SPECIFICATION FOR AN OPEN-ENDED EVOLUTIONARY PROCESS

Perhaps the most important point to arise from the preceding discussion is that processes such as self-reproduction operate *within an environment* rather than in isolation. The properties of this environment, and the ways in which evolving entities may interact with it (and with each other), fundamentally influence the evolutionary process.

Reflecting upon the significance of his work on evolution, and in particular on his demonstration of the possibility of machines that could build modified copies of themselves, von Neumann said: "It is clear that this is a step in the right direction, but it is also clear that it requires considerable additional analyses and elaborations to become really relevant" (von Neumann 1966, p. 131).

It has long been recognized that chief among these additional analyses and elaborations is the incorporation of the evolutionary process into a broader framework that also considers the properties of the environment. Holland has emphasized that the study of adaptation "involves the study of both the adaptive systems and [the] environment. In general terms, it is a study of how systems can generate procedures enabling them to adjust efficiently to their environments" (Holland 1962, p. 299). Moreover, Conrad (1988) stresses that "the characterization of the substrate is of such immense importance for the effectiveness of evolution" (p. 304).

Studies of evolution in vitro, such as Orgel's experiments with evolving RNA sequences using the viral enzyme $Q\beta$ replicase (Orgel 1979), have also demonstrated the need for a better theoretical understanding of these issues. Maynard Smith explains:

> More or less independently of the starting point . . . the end point is a rather small molecule, some 200 bases long, with a particular sequence and structure that enable it to be replicated particularly rapidly. In this simple and well-defined system, natural selection does not lead to continuing change, still less to anything that could be recognized as an increase in complexity: it leads to a stable and rather simple end point. This raises the following simple, and I think unanswered, question: What features must be present in a system if it is to lead to indefinitely continuing evolutionary change? (Maynard Smith 1988, p. 221).

The question raised by Maynard Smith is exactly the one of interest here: What sort of system (in terms of individuals, interactions, and environments) will give rise to an open-ended, and possibly creative, evolutionary process?

1.4.1 Waddington's Paradigm for an Evolutionary Process

A characterization of a process that might be capable of supporting open-ended evolution was proposed by C. H. Waddington 30 years ago (Waddington 1969). He went as far as to call this characterization a new paradigm under which biological evolution should be studied. This paradigm is of particular interest because it provides a general characterization of the individuals involved, of how they interact, and of the kind of environment in which they reside. To my knowledge, little work has been devoted to exploring Waddington's proposal, probably because of the difficulties in capturing it fully with an analytical model (the traditional approach of theoretical biology). However, it is formulated in a way that makes it particularly amenable to synthetic (artificial life) modeling and is therefore an ideal starting place for developing a better theoretical understanding of open-ended evolution within an artificial life framework.

Waddington describes a replicator as "a material structure P with a characteristic Q such that the presence of P with Q produces Q in a range of materials P_i

under circumstances E_j" (*ibid.* p. 115). In other words, Q is the characteristic of a structure P that is inherited when P is replicated—Q is the genetic component of P. The overall scenario is summarized as follows:

> The complete paradigm must therefore include the following items: A genetic system whose items (Qs) are not mere information, but are algorithms or programs which produce phenotypes (Q^*s). There must be a mechanism for producing an indefinite variety of new Q'^*s, some of which must act in a radical way which can be described as 'rewriting the program'. There must also be an indefinite number of environments, and this is assured by the fact that the evolving phenotypes are components of environments for their own or other species. Further, some at least of the species in the evolving biosystem must have means of dispersal, passive or active, which will bring them into contact with the new environments (under these circumstances, other species may have the new environments brought to them). These environments will not only exert selective pressure on the phenotypes, but will also act as items in programs, modifying the epigenetic processes with which the Qs become worked out into [Q^*s]. (Waddington 1969, p. 120).[15]

This general characterization raises some important issues. For example, the requirement that Qs act not only as information but also as algorithms—that they must act as operators as well as operands—locates the relationship between genotype and phenotype at the very heart of the paradigm. (The same requirement was suggested for proto-DNA, in Section 1.3.4.) Waddington points out that the open-ended nature of his model relies on the fulfillment of two conditions: (1) that E_j is an infinite-numbered set and (2) that there are sufficient Qs to provide Q^*s suitable for an infinite subset of E_js.

The first condition is satisfied by the fact that Q^*s are components of E_js. A vital direction for future research is the investigation of the different sorts of ways in which Q^*s could be components of E_js (i.e., how organisms form part of the environment experienced by other organisms), and the evolutionary consequences of such choices.

Of the other condition, Waddington says that

> the second requirement, that the available genotypes must be capable of producing phenotypes which can exploit the new environments, requires some special provision of a means of creating genetic variation . . . It is important to emphasize that the new genetic variation must not only be novel, but must include variations which make possible the exploration of

---

[15] In the original paper, the final word of this paragraph appears as *Q*'s rather then Q^*s. This is likely a typographical error.

> environments which the population previously did not utilize . . . It is not sufficient to produce new mutations which merely insert new parameters into existing programmes; they must actually be able to rewrite the programme (*ibid.* pp. 116–118).

The distinction Waddington is making here is closely related to my distinction between creative evolution and (merely) open-ended evolution. Another important direction for future research is to explore how this second condition can be satisfied.

It is worth mentioning that some existing artificial evolutionary systems, such as Barricelli's studies with evolving game strategies (Barricelli 1963), Conrad and Pattee's model (Conrad and Pattee 1970), and Holland's α-Universes, do have the notion of emergent operators (phenotypes). However, these phenotypes generally have a limited range of action, thereby preventing the systems from engaging in truly open-ended evolutionary processes and restricting their potential for creative evolution.

Now, the requirement in systems capable of open-ended evolution that individual reproducers have selectively significant phenotypic properties, on top of the ability to reproduce, has already been discussed (see Section 1.3.4). However, it may turn out that the fulfillment of Waddington's second condition would require reproducing structures to possess not just one, but *multiple* phenotypic properties, possibly of different functional modalities (e.g., catalysis, light sensitivity, motility, etc.). Maynard Smith has observed that "it seems to be a general feature of evolution that new functions are performed by organs which arise, not *de novo,* but as modifications of pre-existing organs" (Maynard Smith 1986, p. 46). This principle could potentially solve the problem of how individuals can evolve phenotypic structures that utilize new physical modalities (another aspect of my definition of *creative* evolution): a structure with multiple properties (in various modalities) might originally be selected for one of these properties, but it might later turn out (quite accidentally) that some of its other properties also confer (unrelated) adaptive advantages upon the bearer of that structure. In such a scenario, an organism that duplicated this structure might have an adaptive advantage over those possessing a single copy because each structure could be optimized for a single property. In this way, the organism can acquire fundamentally new phenotypic properties. Regarding the issues discussed in Section 1.3.6, note that these considerations would seem to require a notion of materiality rather than a purely formal model.

This perspective may bring some light to bear upon creative evolution, but it also opens up a whole range of new problems relating to the modeling of multiple, and mostly (initially at least) irrelevant, properties of objects. Such questions require much more investigation, but existing work reported in the biological literature on multifunctional enzymes may be helpful (for example, Kacser and Beeby 1984).

1.5 CONCLUSIONS

In this chapter I have discussed the concept of open-ended evolution and the introduction of fundamental novelty during evolution (i.e., creative evolution). Creativity is more subtle than open-ended evolution and involves, for example, the ability of individuals to utilize new physical modalities and to develop new functional relationships with their environment, and for the very notion of individuality to radically change. I have analyzed some existing artificial evolutionary platforms in terms of their ability to support open-ended and creative evolutionary processes. The discussion emphasizes that existing models have generally concentrated on the representation of individuals, and that explicit theoretical considerations concerning the design of the environment (including the issue of how individuals form part of the environment experienced by others, the degree of implicit versus explicit encoding of processes, and the issue of material versus formal models), and of the sorts of interactions allowed between individuals and their environment, have often been lacking. I have also discussed the desirable properties of proto-DNA—a hypothetical structure that might be suitable to act as a seed for an open-ended, and creative, evolutionary process. I suggested that the capacity of this proto-DNA to reproduce should not be easily disrupted by mutations, and therefore that the reproduction process should be implicitly encoded in the environment rather than explicitly encoded on individuals. This led to a discussion of the sorts of phenotypic properties that should be associated with specific proto-DNA structures, on top of their ability to reproduce. In addition, the environment in which the proto-DNA exists should allow unrestricted interactions between individuals, and the representation of individuals should be fully embedded within the arena of competition of the system, so as not to limit the structure's evolutionary potential. I have suggested that the development of material models, as opposed to purely formal ones, may be a useful avenue to explore; in particular, the modeling of matter with phenotypic properties in a number of different modalities. Throughout the chapter I have highlighted various open questions relating to these issues that need to be addressed by future research. In Section 1.4 I described a paradigm suggested by Waddington, which might represent a suitable starting place for a more unified and productive exploration of these issues using synthetic (artificial life) modeling techniques.

ACKNOWLEDGMENTS

I would like to thank John Hallam, Nick Barton, John Holland, Howard Pattee, and Moshe Sipper for useful discussions on the issues addressed in this chapter.

REFERENCES

Adami, C., and C. T. Brown (1994). Evolutionary Learning in the 2D Artificial Life System "Avida." In R. Brooks and P. Maes (eds.), *Artificial Life IV*, MIT Press, pp. 377–381.

Arthur, W. B. (1994). On the Evolution of Complexity. In G. Cowan, D. Pines, and D. Meltzer (eds.), *Complexity: Metaphors, Models and Reality*, SFI Studies in the Sciences of Complexity, vol. 19, Addison-Wesley, pp. 65–81.

Aspray, W., and A. Burks (eds.) (1987). *Papers of John von Neumann on Computing and Computer Theory.* MIT Press.

Barricelli, N. A. (1957). Symbiogenetic Evolution Processes Realized by Artificial Methods. *Methodos* 9:35–36.

Barricelli, N. A. (1962). Numerical Testing of Evolution Theories. Part I. Theroetical Introduction and Basic Tests. *Acta Biotheoretica* 16(1/2):98.

Barricelli, N. A. (1963). Numerical Testing of Evolution Theories. Part II. Preliminary Tests of Performance. Symbiogenesis and Terrestrial Life. *Acta Biotheoretica* 16(3/4):126.

Bedau, M. A. (1998). Four Puzzles about Life. *Artificial Life* 4(2):140.

Cairns-Smith, A. G. (1985). *Seven Clues to the Origin of Life.* Cambridge University Press.

Conrad, M. (1988). The Price of Programmability. In R. Herken (ed.), *The Universal Turing Machine: A Half-Century Survey*, Oxford University Press, pp. 285–307.

Conrad, M., and H. H. Pattee (1970). Evolution Experiments with an Artificial Ecosystem. *Journal of Theoretical Biology* 28:409.

Darwin, C. (1859). *The Origin of Species.* John Murray. (Reprinted in Penguin Classics, 1985. Any page numbers given in the text refer to this reprint.)

Eigen, M., and P. Schuster (1977). The Hypercycle: A Principle of Natural Self-organization. *Die Naturwissenschaften* 64(11):565.

Floreano, D., S. Nolfi, and F. Mondada (1998). Competitive Co-evolutionary Robotics: From Theory to Practice. In R. Pfeifer, B. Blumberg, J.-A. Meyer, and S. W. Wilson (eds.), *From Animals to Animats 5: Proceedings of the Fifth International Conference of the Society for Adaptive Behavior*, MIT Press.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Hillis, W. (1990). Co-evolving Parasites Improve Simulated Evolution as an Optimization Procedure. *Physica D* 42:234.

Holland, J. (1962). Outline for a Logical Theory of Adaptive Systems. *Journal of the Association for Computing Machinery* 9(3):314. (Reprinted in A. W. Burks (ed.), *Essays on Cellular Automata,* University of Illinois Press, 1970. Any page numbers given in text refer to this reprint.)

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Holland, J. (1976). Studies of the Spontaneous Emergence of Self-Replicating Systems Using Cellular Automata and Formal Grammars. In A. Lindenmayer and G. Rozenberg (eds.), *Automata, Languages, Development,* North-Holland, pp. 385–404.

Holland, J. (1995). *Hidden Order: How Adaptation Builds Complexity.* Addison-Wesley/Helix Books.

Hraber, P. T., and B. T. Milne (1997). Community Assembly in a Model Ecosystem. *Ecological Modeling* 103:285.

Kacser, H., and R. Beeby (1984). Evolution of Catalytic Proteins. *Journal of Molecular Evolution* 20:51.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Lazcano, A. (1995). Prebiotic Chemistry, Artificial Life and Complexity Theory: What Do They Tell Us about the Origin of Biological Systems? In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.), *Advances in Artificial Life: Third European Conference on Artificial Life,* Lecture Notes in Artificial Intelligence, Springer, pp. 105–115.

Maynard Smith, J. (1986). *The Problems of Biology.* Oxford University Press.

Maynard Smith, J. (1988). Evolutionary Progress and Levels of Selection. In M. H. Nitecki (ed.), *Evolutionary Progress,* University of Chicago Press, pp. 219–230.

Maynard Smith, J., and E. Szathmáry (1995). *The Major Transitions in Evolution.* W. H. Freeman.

McMullin, B. (1992). *Artificial Knowledge: An Evolutionary Approach.* Ph.D. dissertation, Department of Computer Science, University College Dublin, *ftp://ftp.eeng.dcu.ie/pub/alife/bmcm_phd/.*

Miller, G. F., and D. Cliff (1994). Protean Behavior in Dynamic Games: Arguments for the Co-evolution of Pursuit-Evasion Tactics. In D. Cliff, P. Husbands, J.-A. Meyer, and S. Wilson (eds.), *From Animals to Animats 3, Proceedings of the 3rd International Conference on Simulation of Adaptive Behavior (SAB'94),* MIT Press/Bradford Books, pp. 411–420.

Moore, E. (1962). Machine Models of Self-Reproduction. In *Proceedings of Symposia in Applied Mathematics,* vol. 14, American Mathematical Society, pp. 17–33. (Reprinted in A. W. Burks (ed.), *Essays on Cellular Automata,* University of Illinois Press, 1970.)

Morán, F., A. Moreno, E. Minch, E. and F. Montero (1997). Further Steps Towards the Realistic Description of the Essence of Life. In C. G. Langton and K. Shimohara (eds.), *Artificial Life V: Proceedings of the Fifth International Workshop on the Synthesis and Simulation of Living Systems,* MIT Press, pp. 216–224.

Muller, H. J. (1966). The Gene Material as the Initiator and the Organizing Basis of Life. *The American Naturalist* 100(915):517.

Myhill, J. (1963). The Converse of Moore's Garden-of-Eden Theorem. In *Proceedings of the American Mathematical Society,* vol. 14, pp. 658–686. (Reprinted in A. W. Burks (ed.), *Essays on Cellular Automata,* University of Illinois Press, 1970.)

Nuño, J. C., P. Chacón, A. Moreno, and F. Morán (1995). Compartmentation in Replicator Models. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.), *Advances in Artificial Life: Third European Conference on Artificial Life,* Lecture Notes in Artificial Intelligence, Springer, pp. 116–127.

Orgel, L. E. (1979). Selection in Vitro. *Proceedings of the Royal Society, London,* B, 205:442.

Packard, N. H. (1988). Intrinsic Adaptation in a Simple Model for Evolution. In C. G. Langton (ed.), *Artificial Life,* Santa Fe Institute Studies in the Sciences of Complexity, vol. 5, Addison-Wesley, pp. 141–155.

Pattee, H. H. (1988). Simulations, Realizations, and Theories of Life. In C. Langton (ed.), *Artificial Life,* Santa Fe Institute Studies in the Sciences of Complexity, vol. 6, Addison-Wesley, pp. 63–77. (Reprinted in M. A. Boden (ed.), *The Philosophy of Artificial Life,* Oxford University Press, 1996.)

Pattee, H. H. (1995a). Artificial Life Needs a Real Epistemology. In F. Morán, A. Moreno, J. J. Merelo, and P. Chacón (eds.), *Advances in Artificial Life: Third European Conference on Artificial Life,* Lecture Notes in Artificial Intelligence, Springer, pp. 23–38.

Pattee, H. H. (1995b). Evolving Self-Reference: Matter, Symbols, and Semantic Closure. *Communication and Cognition—Artificial Intelligence* 12(1–2):28.

Pesavento, U. (1995). An Implementation of von Neumann's Self-Reproducing Machine. *Artificial Life* 2(4):354.

Ray, T. (1991) An Approach to the Synthesis of Life. In C. G. Langton, C. Taylor, J. D. Farmer, and S. Rasmussen (eds.), *Artificial Life II,* Addison-Wesley, pp. 371–408.

Ray, T. (1994). An Evolutionary Approach to Synthetic Biology: Zen and the Art of Creating Life. *Artificial Life* 1(1/2):226.

Ray, T. (1997). Evolving Complexity. *Artificial Life and Robotics* 1:26.

Rocha, L. M. (1998). Selected Self-organization and the Semiotics of Evolutionary Systems. In S. Salthe, G. Van de Vijver, and M. Delpos (eds.), *Evolutionary Systems: Biological and Epistemological Perspectives on Selection and Self-Organization,* Kluwer Academic, pp. 341–358.

Ruiz-Mirazo, K., A. Moreno, and F. Morán (1998). Merging the Energetic and the Relational-Constructive Logic of Life. In C. Adami, R. Belew, H. Kitano, and C. Taylor (eds.), *Proceedings of Artificial Life VI,* MIT Press, pp. 448–451.

Schmitz, O., and G. Booth (1996). Modeling Food Web Complexity: The Consequence of Individual-Based Spatially Explicit Behavioral Ecology on Trophic Interactions. *Evolutionary Ecology* 11:398.

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. In R. Brooks and P. Maes (eds.), *Proceedings of Artificial Life IV,* MIT Press, pp. 28–39.

Szathmáry, E., and L. Demeter (1987). Group Selection of Early Replicators and the Origin of Life. *Journal of Theoretical Biology* 128:486.

Taylor, T. (1997). *The COSMOS Artificial Life System.* Working Paper 263, Department of Artificial Intelligence, University of Edinburgh, 1997. Available from *http://www.dai.ed.ac.uk/daidb/people/homes/timt/papers/.*

Taylor, T. (1999). *From Artificial Evolution to Artificial Life.* Ph.D. dissertation, Division of Informatics, University of Edinburgh.

Turing, A. (1936). On Computable Numbers, with an Application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society, Series 2,* 42:265.

Van Valen, L. (1973). A New Evolutionary Law. *Evolutionary Theory* 1:30.

von Neumann, J. (1966). *The Theory of Self-Reproducing Automata.* University of Illinois Press.

Waddington, C. H. (1969). Paradigm for an Evolutionary Process. In C. H. Waddington (ed.), *Towards a Theoretical Biology,* vol. 2, Edinburgh University Press, pp. 106–128.

Zaug, A. J., and T. R. Cech (1986). The Intervening Sequence RNA of *Tetrahymena* Is an Enzyme. *Science* 231:475.

2 CHAPTER

Recognizability of the Idea: The Evolutionary Process of Argenìa

Celestino Soddu Milan Polytechnic

2.1 INTRODUCTION

"Argenìa" is the new word that I have coined for morphogenetic, generative, and evolutionary design, or more specifically, for my personal and subjective approach to design. My works are not tools for designers, but design products realized with subjective and original software.

My idea is that a generative approach can realize operative *metaprojects.* These could also be called "idea-products": software that is able to generate an endless sequence of products. The idea-projects create a new market: an industry can buy a morphogenetic idea-project for lamps, for example, and use the endless sequence of automatically generated 3D models to produce different lamps every time (another example might be an idea-project used as an autoreprogramming tool for robots). Customers can choose their unique object by activating, on the Internet, a generative tool and sending their request to the industry. Or perhaps a mayor could order the idea-project of evolution[1] of his or her town and use it to control possible new scenarios and changes of the environment.

What I call the "recognizability of the idea" is the first step toward the possibility of selling the idea-product, performed as a generative project, as artificial DNA.

---

[1] "Evolution" is used here to mean increasing complexity.

2.2 RECOGNIZABILITY, IDENTITY, AND COMPLEXITY

We build up our recognizing codes since infancy. We learn, from birth, to recognize everything in the world, to recognize a chair among other similar objects. We become able to build our own peculiar idea of a chair. When we are in front of a chair that we have never seen before, even though this shape is new and different from any other known chair, we are still able to recognize it as a chair. Likewise, if we are in front of a painting by van Gogh that we have never seen before, we can recognize it as a painting by van Gogh.

Recognizability is, therefore, our abstract synthesis of past moments, which allows us to evaluate unpredictable new events. But recognizability is not an objective concept. We cannot learn to recognize objects outside our experience. The synthesis is subjective; it is a distinguishing mark of our uniqueness and the identity of human beings. It is the subjective building of categories, of *species* for the identification of an individual object.

This synthesis is also the first step in the design process.

Design can be regarded as a natural or artificial dynamic system that designers try to forge upon reality by drawing a model of possible desirable events and shaping this model through one's own thoughts and wishes.

I define generative design as a type of evolutionary process that is able to generate a sequence of results, where each result is different, but recognizable as belonging to a species. For this reason a *control structure* of recognizability becomes necessary in this approach. Such a structure is fundamental in building generative projects, in controlling evolutionary processes, and in controlling the results (the species). The recognizability of the idea (and of the species, which I consider to be one of the possible representations of the idea) is peculiar to the field of generative design. In this field the hand of the designer defines the design work.

In my work these considerations are never merely theoretical reflections. Every hypothesis, every possible path of development that I propose, every logical approach to design has been operatively experimented and emulated as evolutionary design with suitable software.

I have performed my generative projects motivated by the need to allow, without preclusion, *direct access to the complexity of a natural/artificial environment,* to the dynamics of the unpredictable alchemy of increasing complexity, in the growing of evolutionary processes in architecture, industrial design, and art. The goal is to identify and approach *creativity as the clock of evolution.*

2.3 EVOLUTIONARY CODES: ARTIFICIAL DNA

I regard the software I have designed as a "designer of species"—we can use it as artificial DNA to generate a multiplicity of possible events. My first experiments were in the field of town environment and architecture. This software is the representation of my subjective approach to town environment, of my idea of architecture (the result of over 30 years of experience as an architect).

The generative software is based on a presumed homology between natural and artificial spheres, and their belonging to the world of chaotic systems. Following the possible similarities between natural and artificial evolution systems, it seems likely that, like DNA in nature, we can enable the morphogenic and generative projects of "Argenìa" to have considerable powers. For example, Argenìa can be made to control and increase evolution processes, define the complexity of possible meanings/functions, the stratification of multiple orders, the mode of contamination between orders and between different disciplinary fields, and affect the recognizability of every result the system can generate. Importantly, human/machine interaction allows us to manipulate the development of this project—operating upon the parameters of the evolution code and not just changing the final arrangement.

There are two important topics in the design of this tool: complexity and the relationship between species and individual.

To manage the complexity I used the idea that complexity cannot be generated ex novo, but only via a process to stratify sense into a flowing simulation with a temporally irreversible path. We can activate and control this stratification if we design a system that uses a self-organizing paradigm, capable of increasing its identity and recognizability as time progresses. To build this paradigm I made use of chaotic dynamic systems controlled by algorithms (which means that they never produce the same result twice).

I have used a fractal but nondeterministic logical frame. In other words, every design decision cycle has other decisions nested inside using a lot of other cycles, and so on. The structure shows self-similarity—the cycles are, as in fractal objects, always the same. Differences and unpredictability are born from resonance with other cycles, from the time of activation, and from an ever-changing flow of information.

Each cycle represents the whole structure of a simulation of design decision choices. It transforms the results into possible shapes. This method is designed by the following:

1. The use of a paradigm to control the self-organization procedures. A tool represents and controls how complexity is gained in designs and also

represents the adaptability to future developments. This is the method that allows us to reply to an answer by putting one of the possible formal matrices (see number 3) into the paradigm.

2. The identification and sharing of the random margins between answers and shaping replies. The system uses and represents these margins as "operable fields" for the design choices to improve the project evolution.
3. The set of possible formal matrices that define abstract shapes, used in the giving body to enable a set of possible performances. These formal matrices are not a database. They are extemporarily generated by the bound-up cycles, by a set of simultaneous devices operating on a series of different fields such as geometry, dimension, materials, technology, complexity, and so on.

At the end of every cycle (and of the related and multiple progressive nestings), the result is an increased complexity of form, and the related passage into a more evolved representation of needs. New requirements are produced, as every generated result was also born because of a subjective and random approach. For example, a shape used in a design may not have been necessary before the choice, but it became necessary after, and as a part of the project, it generates further requests. This also happens if we later remove it because we consider this result to be obsolete. The result is placed into the project history, the needs generated by this result have been satisfied, and we can appreciate its contribution as a patina over time. Patinas measure the complexity gained and the growth of the specific identity of the project, shaped by the past research, and can be used as training events (Soddu and Colabella 1992).

2.4 NATURAL/ARTIFICIAL COMPLEXITY

Towns evolve with time, expanding, contracting, transforming themselves, their image, and their characteristics under economical, cultural, political, and technological pressures. It is possible to predict the development of towns in the short term, as for any natural system. Everything happens according to precise rules derived from urban planning, current technology, and economical and social influences.

However, predictions are almost impossible in the long term since towns are inhabited by human beings and each choice, each successive step brings about margins of subjectivity and unpredictability. This subjectivity produces one of the most important characteristics of urban shape: formal diversity of solutions responding to the same situations.

It must be remembered that the most subjective and casual aspects of decision making have little influence and will disappear in the subsequent transformations of a town. However, some events or marginal aspects will persist and, because of accompanying casual circumstances, they will influence the course of future development more and more. This influence, once it has been consolidated, becomes irreversible and can outgrow the importance of the event that caused it.

Some architectural forms, originating from accidental, casual, or subjective choices, very often lose their raison d'être and become pure formalization of relations between following events; they often disappear, surviving only as a trace, as a testimony of the passage of time, leaving a strong characterization of urban shape.

Piazza Navona, in Rome, is a good example of this continuity. The form, the dimension, the sequence of the various architectural structures of Piazza Navona have been determined by a building no longer in existence and for which little is known about its past function. This building has had an irreversible influence since its form has conditioned the surrounding architecture and has contributed to the formation of one of the most fascinating places of Rome.

Other Roman buildings, from the same period, conceived then to have a greater influence on urban development, as for example Trajan Forum, did not have the same fortune.

Of the many fragments breaking off from a glacier, only a few, at random, cause an avalanche, and only a few avalanches can modify the environment. Only a few forms in the environment have the capacity of fascination due to the chance contribution of rivers, winds, vegetation, and—why not—architecture.

Such events are not predictable, for the succession of ever-growing changeable events cannot be predicted, and above all, because the unplanned, subjective event that may become incisive on the whole system cannot be imagined and be clearly defined in advance. The system is not predictable; however, it can be represented through numerous simulations done in order to produce a large variety of possible results.

My research introduces mechanisms/software for the simulation of time in a town system whose instability is due to the complexity of objectives that are often contradictory. This mechanism should produce an endless series of urban events temporarily parallel and recognizable as different "individuals" from the same "species."

This implies that there is no possibility of identifying and defining an order capable of assuring, with time, an urban characteristic, if this order takes form from a static evaluation, from the analysis of one moment of equilibrium and leaving out the dynamics of time. Evolution, and also time, starts from the instability of the system.

Urban growth can therefore be more appropriately analyzed if it is represented not as a temporal stratification of successive equilibria but as an unstable system that proceeds through irreversible changes—even if it has a modus operandi that can be individualized and characterized. This instability, once it has been represented mathematically, is characteristic of an unpredictable dynamical system.

Towns, in other words, can be considered as complex entities that can be individualized in their uniqueness, and above all as evolving entities capable of modifying their form in an unpredictable way.

Evolution (the passage through irreversible events) and morphogenesis is characteristic not only of urban shape but also of living beings. Notwithstanding their artificiality, towns are not very different from nature in this respect and, according to the trend of current scientific debates, may even be considered as living entities. This evolution can be emulated using methods from artificial life.

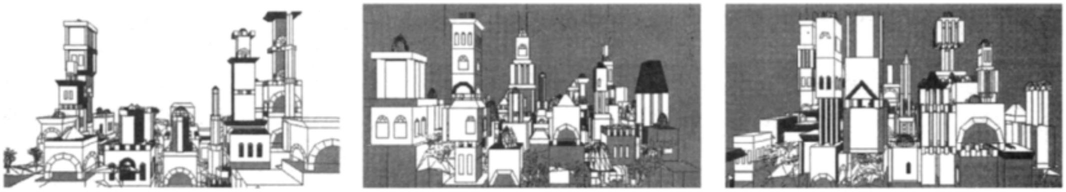
2.5 GIOTTO, A MEDIEVAL IDEA IN EVOLUTION

The results of my first generative experiences in town design were very successful, and my generative projects have improved in their complexity with progressive stratification of new ideas over 10 years, performed by evolutionary methods inside the software.

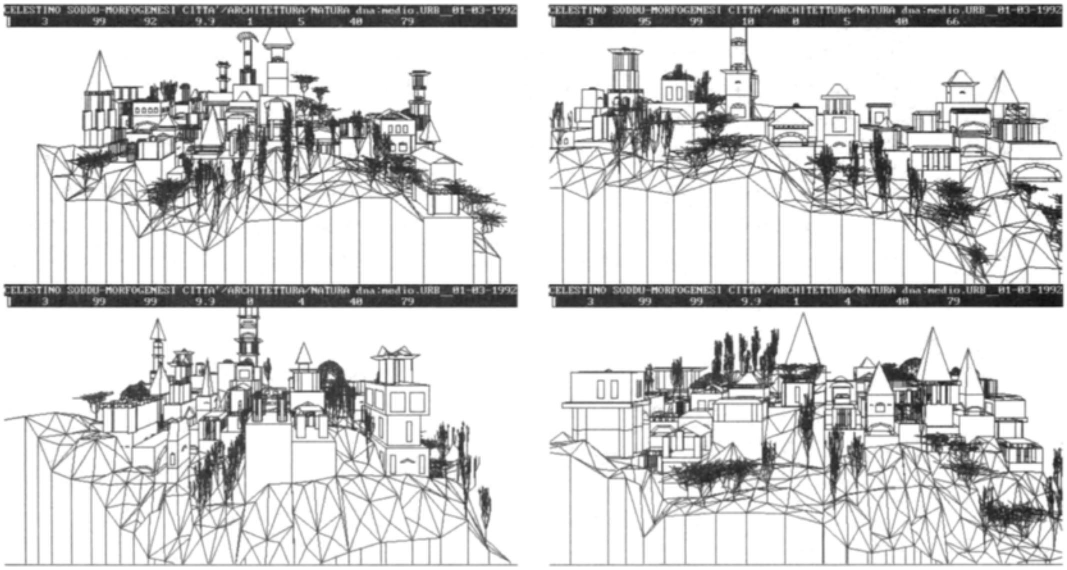
The first version, realized in 1988, is evolutionary software that emulates the increasing identity and recognizability of some particular town environment existing in Italy.

I designed, as a natural species, the DNA of a typical Italian medieval town. I referred, for this experimentation, to the drawings of Giotto, and I interpreted these drawings as an evolving idea of a town environment. To obtain an acceptable complexity of the urban image I simulated and ran linear and nonlinear dynamics of evolution of this type of urban image. Following my subjective interpretation as an architect, I proceeded to identify and discover the rules of the game, the modes of the transition between one order and another, the role of randomness in increasing complexity and the power of time in shaping the environmental image.

So I produced (crafting it as original software, built like DNA code) the first experimental generative system, designed following and representing the



2.1 FIGURE Generated environment of Italian medieval towns.


CELESTINO SODDU-MORFOGENESI CITTA'/ARCHITETTURA/NATURA dna:medio.URB_01-03-1992
3 99 92 9.9 1 5 40 79
CELESTINO SODDU-MORFOGENESI CITTA'/ARCHITETTURA/NATURA dna:medio.URB_01-03-1992
3 95 99 10 0 5 40 66
CELESTINO SODDU-MORFOGENESI CITTA'/ARCHITETTURA/NATURA dna:medio.URB_01-03-1992
3 99 99 9.9 0 4 40 79
CELESTINO SODDU-MORFOGENESI CITTA'/ARCHITETTURA/NATURA dna:medio.URB_01-03-1992
3 99 99 9.9 1 4 40 79

2.2 FIGURE Generated scenarios of Italian medieval natural/artificial environment.

dynamic evolution of urban image and identity. Every time this system is used, many 3D models of completely unique virtual medieval towns can be generated (see Figures 2.1 and 2.2, and Color Plate 1).

Every model is one of the possible parallel histories of "my" medieval town evolution. The difference between one and another virtual model is like the difference between one and another individual belonging to the same species—a difference shaped by the random components of the evolution. But every town image is characterized, regardless of the differences, by the morphogenetic nature that identifies this particular design approach to the evolutionary code: my design idea of the medieval towns came from my interpretation of Giotto's frescoes. To put it another way, I mean that every time we use this software, we run through one of the possible town histories. And every history is always different, but it is also parallel to the others. Like in every fable, the structure is always the same, but every time we tell it, we run one of the possible subjective paths (Soddu 1992). And the possibility of identifying the character of these towns is linked to the management of increasing complexity because complexity is the representation of the laws of evolution.

The real problem is the management of this complexity. The complexity derives from evolution, from having passed through a "history" that has enchanted the identity of the system. Most of the time, the credibility and recognizability of an urban environment is removed by simplification—and typical optimization is a simplification.

Complexity involves the ability to adapt to each possible consumer, the progressive stratification of significant structures that, reacting to the subjective choices of each individual, shows unexpected results (but still relevant to the requirements). And these answers, unexpected and subjective, can make our cities livable (Soddu and Colabella 1995).

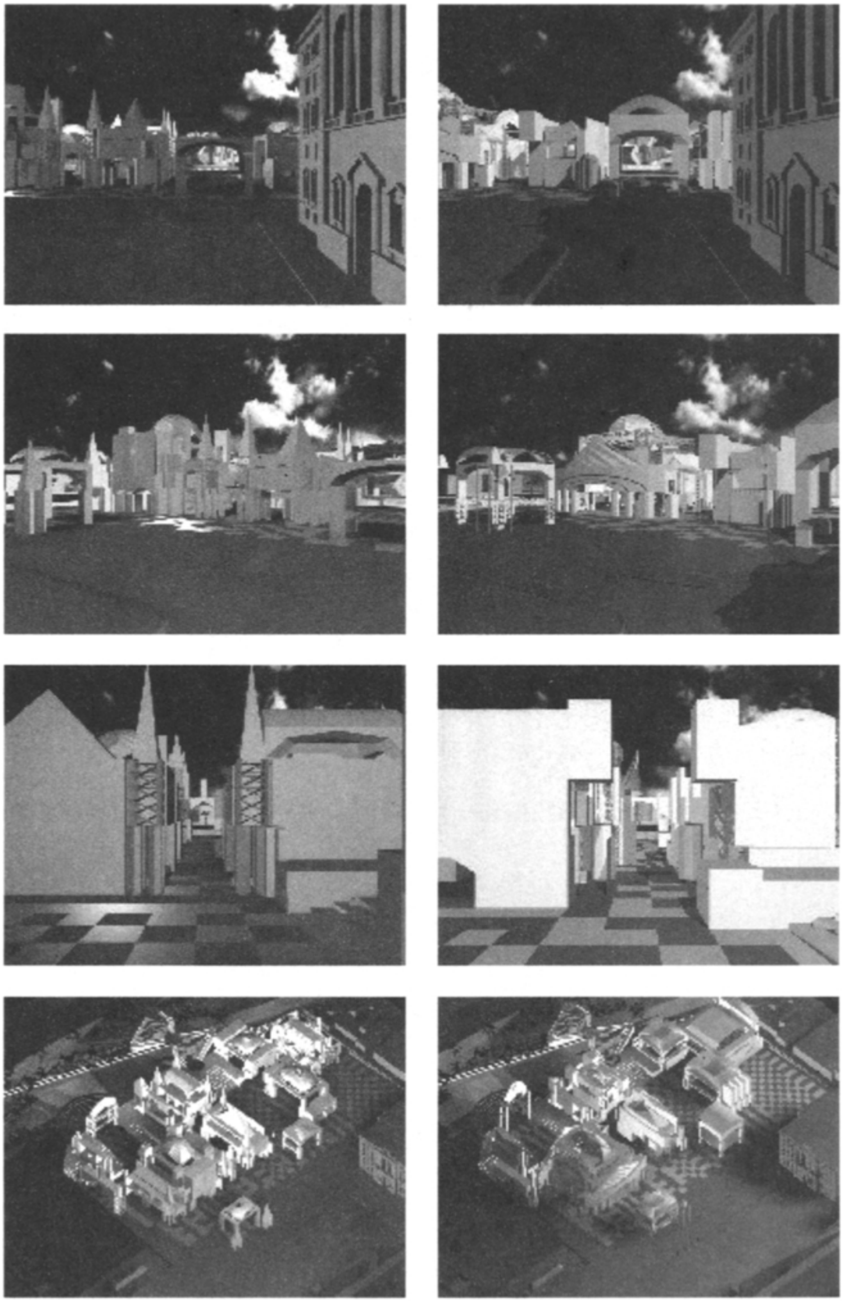
2.6 ROME, FUTURE SCENARIOS

An example of the professional use of this environment generative software was the project of the evolution of Borghetto Flaminio in Rome (see Figure 2.3 and Color Plate 2).

The evolutionary code of this environment was defined and a sequence of possible scenarios was generated. In the images, two different scenarios generated by my evolutionary software (the last one is on the right) are shown from the same point of view.

2.7 BASILICA, GENERATIVE SOFTWARE TO DESIGN COMPLEXITY

Basilica (first version in 1990) is a project able to generate an endless sequence of architecture, all different but all belonging to the same idea. It is a work in progress, intended to gain an increased recognizability of my ideas of architecture, starting from classic and renaissance paradigms and controlling the



2.3
FIGURE

Generated Roman environment (two steps).

evolutionary process through methods from artificial life. It may also be a tool to manage the increasing quality of architectural projects, enabling possible different paths of architecture to be formalized.

In the design process every choice is, in effect, a moment of formalization. The options are formal options, and the choice of a formal option is conducted through the preview of new virtual environments. And so these virtual worlds grow up.

When the choice of a design is made, the formalized event is added to the design paradigm and the global environment is evaluated with respect to the possible worlds that could have been improved in the meantime. In other words, as the project develops, gaining new shapes and new events, the reference virtual worlds—the thinkable worlds—gain in evolutionary structuring and lead to possible desirable scenarios. Managing this evolution, this dynamic accumulation of significance, we can get the tool to measure the correspondence between subjective posture and intersubjective imagery, that is, the imagery shared by a lot of different individuals in order to measure the quality as a value that must be achieved to generate acceptance.

To manage this procedure there are three necessities. The first is the possible worlds, the virtual environment we use in getting a dynamic evolution in affinity with the process of sliding between subjective and intersubjective spheres. This means that dynamic evolution enables increasing complexity, an increase of the possible significance, a complex result that is closest to an objective excellence defined as the most pertinent reply to ever possible and random subjective approaches. To gain the objective spheres is to gain a universe of possible subjective spheres.

Second, every formalization choice we make in a requirement/response cycle must be proposed again in the next cycle as a new requirement. With this procedure we can, during the designing, gain two targets:

- To enable the accumulation of functional and possible subjective signification, and at the same time to distinguish in our project the categorical events from events that cannot escape from a hard subjective point of view.
- To build a logic structure to begin a formalization that also allows exceptional events. Exceptions are necessary to enable the paradigm jumps that construct the multiplicity of possible virtual scenarios. But, to do that, every exception must be used as a new question in the next step of the design procedure.

Third, the sequence of formalization cycles must generate a dynamic evolution of possible shapes, of possible architectural scenarios. This evolution is directly connected with the information and complexity. The number of possible alternatives, of possible scenarios is, in fact, the measure of the resources of the designed environment to reply to the possible needs. It is not the measure of the quality, but it is certainly a good key for evaluating such quality.

The generative software Basilica that I have designed and produced was created to operate inside different real/virtual environments and differently designed results/desirable worlds. Basilica can show the multiplicity of possible scenarios that every design choice can generate. It is an approach that generates unique possible scenarios for every single composition idea, even for a single new design choice. Such results allow us to evaluate the quality of the design process in action.

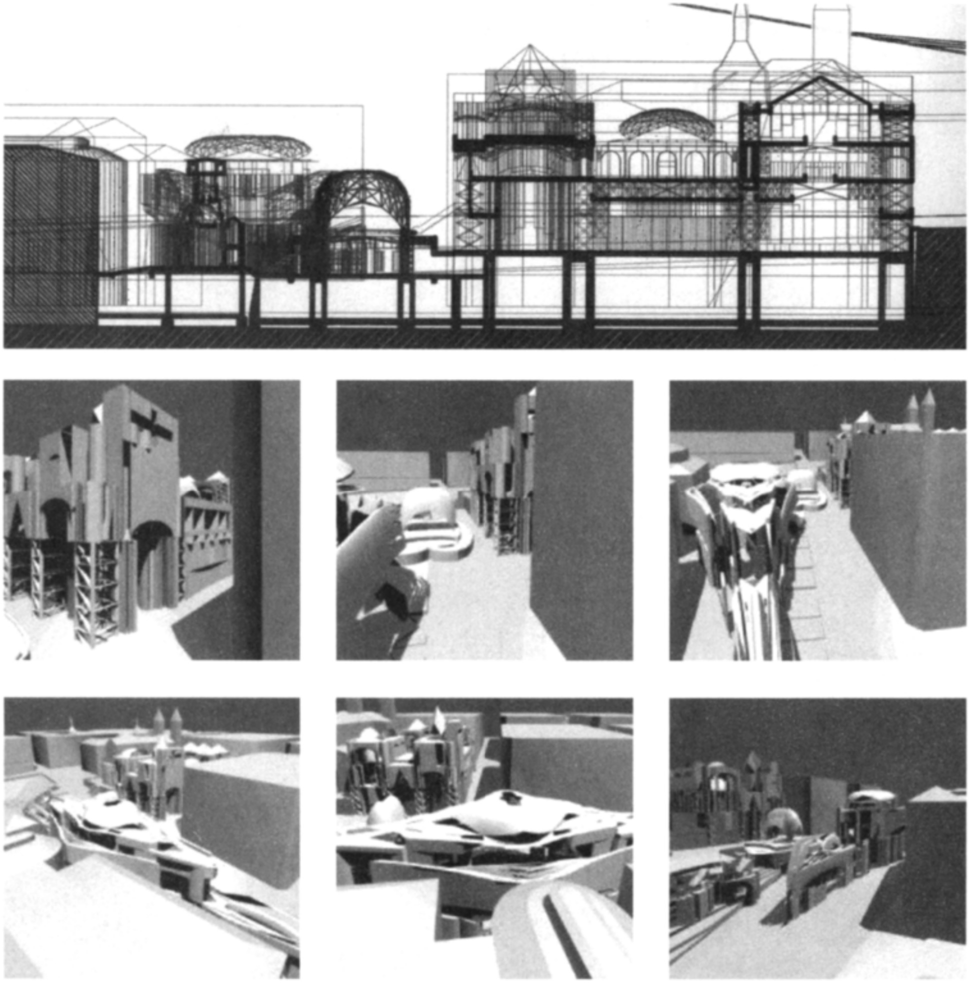
Basilica gives us a concrete representation of the evolution dynamics of design ideas through continuously increasing complexity of virtual environments. In other words, the main contribution of Basilica is in the explicitness of the dynamic evolution of the project from subjective spheres to intersubjective ones, allowing evaluation of increasing quality during the design process.

Using Basilica I have drawn some conclusions. Because this project enables the generation of many different 3D scenarios as a projection of a single composition idea, it seems that the multiplicity of possible shapes does not concur with creativity. It is only a possible representation of the idea, as logical and formal DNA—a postmetaphysical structure of the same idea. This reflection shows the importance of any choice operated inside the evolutionary logic, of the tools that allow these choices, and of the evaluation and control of the idea before its infinite possible realizations inside a shape (Soddu 1993).

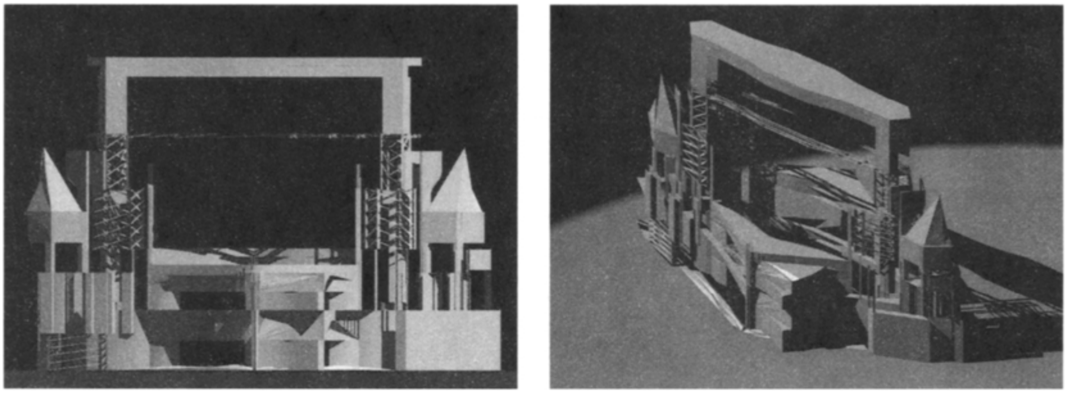
2.8 MADRID AND MILAN, GENERATED ARCHITECTURE

An example of the professional use of Basilica is the project of the enlargement of the Prado Museum in Madrid (Figure 2.4), performed as an evolutionary process of a micro town environment. It was realized with original software representing our design idea of evolutionary codes of Madrid.

The multimedia urban stand (Figure 2.5) and the multimedia square in Milan (Color Plate 3) are architectural projects completely generated using Basilica and its interface to control the paradigm of evolution for the particular requests for each project.



2.4 FIGURE Enlargement of the Prado Museum, Madrid (generated final scenario).



2.5 FIGURE Multimedia urban stage in Milan.

2.9 ARGENÌA, THE NATURAL INDUSTRIAL OBJECT, AND THE ARTIFICIAL UNIQUENESS OF SPECIES

Argenìa—the generative design of industrial objects—is the approach used to build the DNA of artificial articles and to produce, with present industrial devices, unique objects, each one different but recognizably derived from the same idea, just as natural articles can be recognizably members of a species.

Before the industrial era each object was unique, unrepeatable, and strongly connected to the identity of a person. This bond, together with the uniqueness and unrepeatable property of the object, determined one of the qualities of artificial objects, a value above any intrinsic value, quantified by the materials and by the execution. The result was, for the object, an extremely slow obsolescence related to the functionality (also aesthetic functions) and not related to a future "ultimate model."

During the two centuries of the industrial era, now over, the objects were mass-produced. From the assembly line, identical products were obtained, and this uniformity was celebrated, often overestimating the processes of optimization and building an aesthetic of repetition. But this approach was based on some presuppositions that today have lost their validity and could be disproved easily for the reason that they no longer respond to the potential and expectations of human beings:

Objects realized in series cost less than objects that are different and unique. Production using digital control can realize, with the same operational costs, unique objects or repeated objects. A printer, for example, costs the same whether it prints 10 different pages or 10 versions of the same page. The cost difference belongs to the commands (in our example, the text file), to the reprogramming actions in the robot, for the design.

If the design is an operative metaproject, it will be able to emulate the process to generate possible results—and to generate these results as they are in reality, as ever-different scenarios. An operative metaproject can realize these scenarios, formalizing them like real-time reprogramming actions of the digital control machines, of the robots. In this case, the additional cost, if it exists, is due only to the design operations.

Optimization of the functions made it necessary to identify a "unique" design result. The misconception of product optimization is over. We cannot identify a design result as the only one "necessary" once we have discovered or rediscovered the role and the irreplaceable importance of subjectivity of the designer.

Design is not an inferential process. Identification between function optimization and the design of a single object is not thinkable. A full definition of the functions must always incorporate margins of variability of the formal matrices, of the technological matrices, and of materials. Mass production of identical objects is therefore an impoverishment without benefits of one of the final qualities of the object. Mass production cannot reach the possibility of linking objects to different human beings, and to their diverse requests. Mass production reduces a fundamental function that qualifies the object: the capability to increase the identity and the uniqueness of each human being.

The quality of the designer and the quality of a particular design idea produces the final result, the crystallized scenario of the last action. This is the only possible result with respect to the design idea; it is the only possible realization.

The quality of a project cannot be reduced into a single final result. In order to show and evaluate the quality of a project, it is no more acceptable than the action of rebuilding the project, a posteriori. It is no more acceptable starting from the result in order to demonstrate that the same result is the only possible result if a determined quality was wanted. If we want to clarify the relationships between design, imagination, and creativeness, we must identify inside the design processes what is possible to emulate using computers and what is, instead, the exclusive dominion of the human mind—what is not and could not be emulated using machines. The idea, as a subjective construction of a hierarchy of possible relations and inferences about a future object, cannot be emulated by a computer for the reason that an idea is not the fruit of inductive or inferential processes, but of processes of adduction—that is, of interpretative processes that strongly belong to a subjective approach. Once conceived, the idea could be explained and communicated in two ways: with a series of projects or with a subjective metaproject.

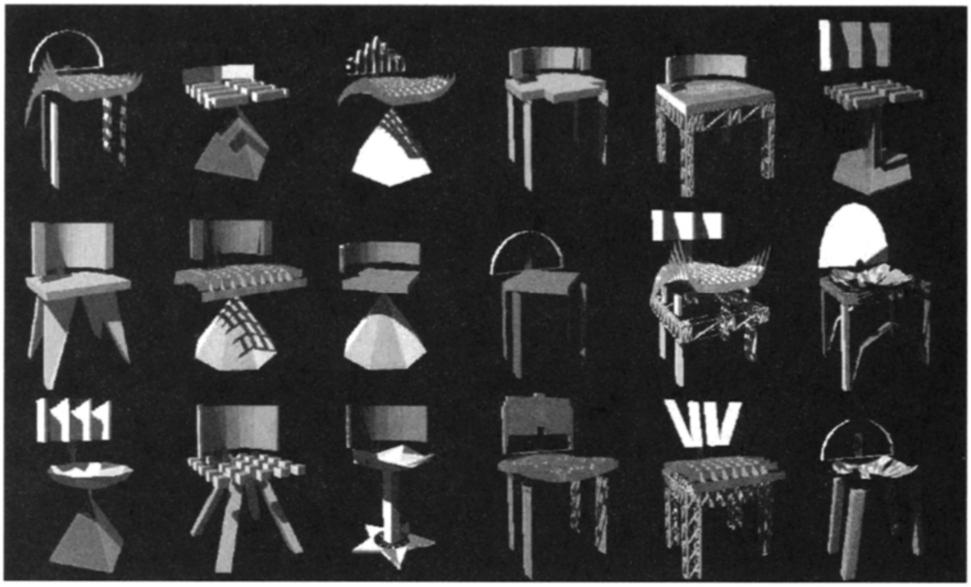
But a series of projects is not an exhaustive exploration of the idea; it carries out only some of its possible scenarios. And the process of building a scenario is a process that could be emulated by a computer because it uses processes of inferential synthesis, using procedures of artificial intelligence.

If, instead of explaining the idea through some results that characterize only some of the possible scenarios, we carry out a subjective metaproject, we have carried out a total communication of the same idea. A subjective metaproject is a computer program of artificial intelligence that explains the idea, since it is able to emulate the processes of building scenarios with the computer and to manage these scenarios in the manufacturing sequence. Such design is *Argenìc design.*

At this point, if we exclude the relative preconceptions about cost, about function optimization, and about recognizability of the design, Argenìc design is a conceptual and efficient innovation created to enable the realization of products for the third millennium—unique and unrepeatable products, like the objects the human beings have always made, but realized by industry (Figure 2.6). These objects are made in the measure of human beings because they fit the strong subjective approach of the user. These products are good for the environment, not only because they may be recyclable, but because they have a slow obsolescence (Soddu and Colabella 1997).

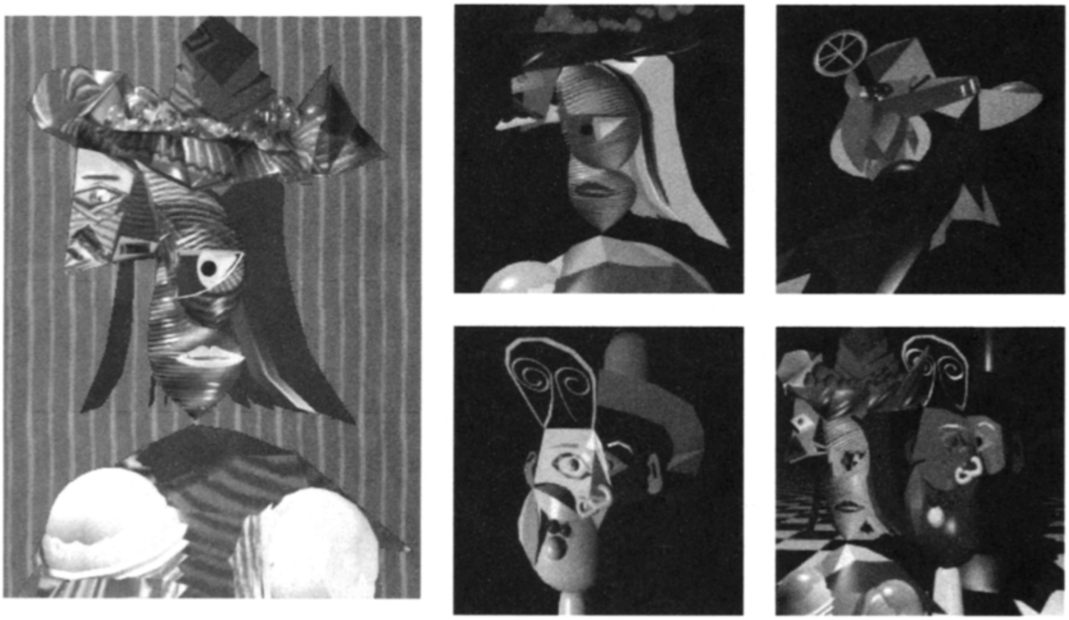
2.10 ARGENÌC ART: PICASSO

Argenìa from Picasso (Figure 2.7) was an experiment done using evolutionary methods to increase and control recognizability of Picasso's woman portraits. As Picasso repainted Velasquez, I tried to repaint Picasso with a generative art project capable of generating a sequence of Picasso's woman portraits, each one different but recognizable as Picasso, but also belonging to my interpretation of these portraits. As happens in all cultural activities, different identities are stratified in the same object.



2.6
FIGURE

A sequence of “unique” chairs generated with Argenìa.



2.7
FIGURE

Repainting Picasso with Argenìa.

2.11 CONCLUSIONS

Building and reading recognizability in evolutionary processes needs a *subjective approach.* The ability to recognize an object belongs to the systematic explanation of one's own subjectivity and identity for unpredictable events. The identity of each designer is the key to the synthesis of the structure of recognizability.

The idea is a hypothesis of the process, not a preview of the result. It may be a preview of the character of the plurality of possible events. Recognizability is strongly connected with the idea and may be controlled through the laws of evolution. These laws are subjective hypotheses of "how" the system evolves and may be represented using a paradigm that defines the progression of auto-organization and reciprocal contamination among events. These laws, and the related algorithms, because they are operative hypotheses, cannot be interpreted using an analytical approach; they can only be verified a posteriori. Once performed, the paradigm becomes a metaproject that can generate an endless sequence of different scenarios all belonging to the idea; for example, see Figure 2.8.

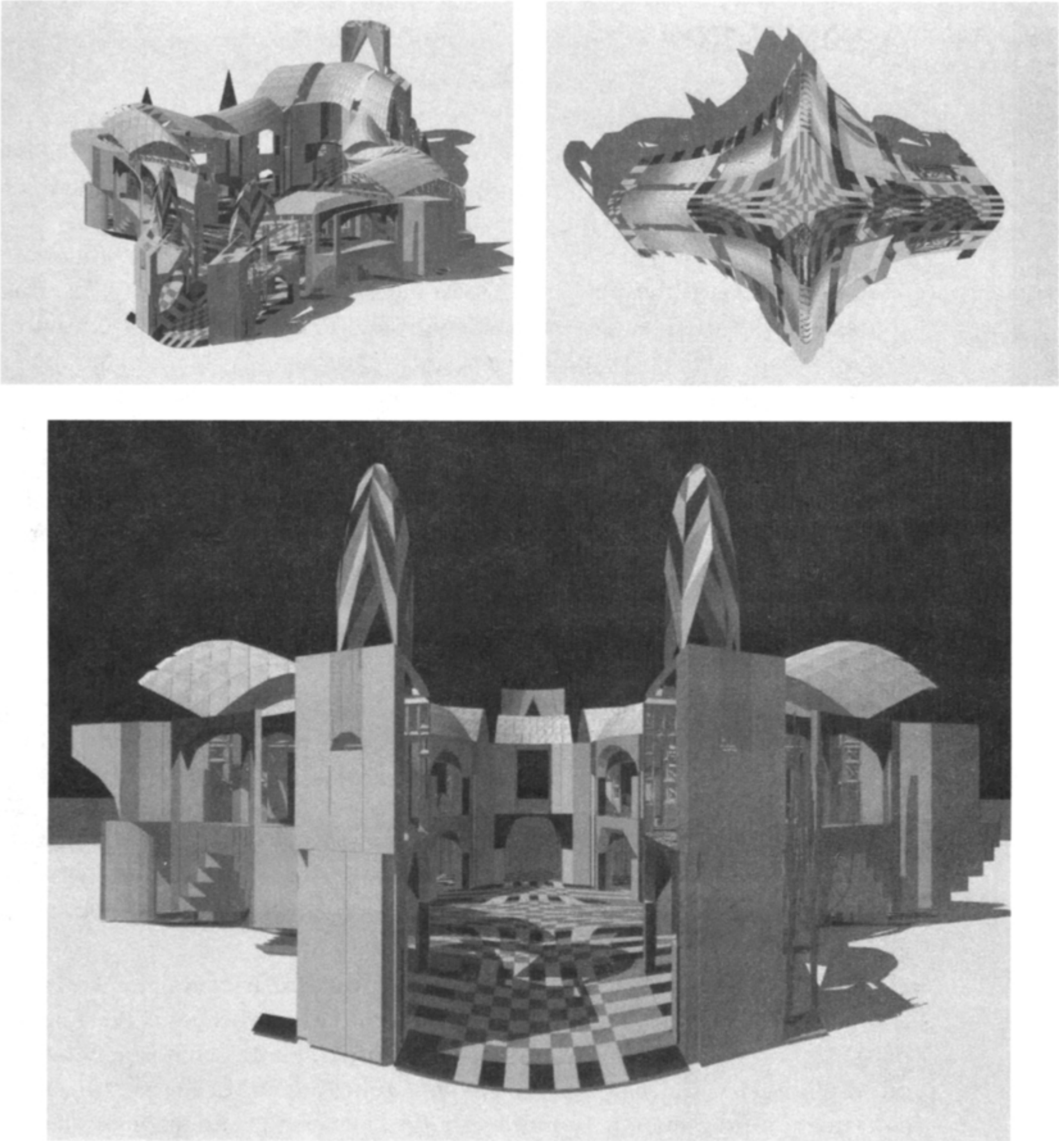
And, finally, a question: Can we use the beauty of generated results as measure of their quality? We all think that natural objects are normally beautiful.

According to Leon Battista Alberti, beauty is harmony, proportion, and logic among all parts of an object. We can identify this proportion and logic with the representation of its evolution through time. Can this complexity, gained with the artificial life of an evolutionary process, answer to the unpredictable aesthetic requests of the user?

The dynamics of beauty have infinite possibilities. The abstraction from an unknown future together with the ability to obtain pertinent answers to possible and unpredictable incoming requests, and ambiguity as multiple possible points of view, are the characteristics of beauty.

In a generative project (i.e., the use of evolutionary laws to increase the recognizability of each generated and different result), beauty is the final and most important test. The beauty of an unpredictable event that is generated by our evolutionary project and is, at the same time, unequivocally recognizable as a pertinent representation of our design idea, may be the endorsement of a well-performed and complex evolutionary system.

And it may also be the confirmation that *identity and subjectivity are strongly connected to the evolutionary structure of design.* Identity and subjectivity define the developing field of quality and explain why we cannot think of a work inside the evolutionary processes in design preferring analytical/objective paths and not a path that began with subjective hypothesis, as scientific research does.



2.8
FIGURE

Generated architecture: a new urban space in Milan, the "Caravanserraglio."

REFERENCES

Soddu, C. (1979). *L'artificiale progettato* (The Designed Artificial Ware). Casa del Libro Publishing.

Soddu, C. (1987). *L'immagine non euclidea* (The Non-Euclidean Image). Gangemi Publishing.

Soddu, C. (1989). *Citta' Aleatorie* (Random Towns). Masson Publishing.

Soddu, C. (1992). The Design of Morphogenesis. To Design the Genetic Code of Artificial Environment. Museum of Design, Zurich, Jan. 26, 1992.

Soddu, C. (1993). The Morphogenetic Design as AI System to Support the Management of the Design Processes through the Total Quality of the Built-Environment. In *The Management of Information Technology for Construction, First International Conference.* Singapore, Aug. 17–20, 1993.

Soddu, C., and E. Colabella (1992). *Il progetto ambientale di morfogenesi: il DNA dell'artificiale* (Morphogenetic Environmental Design: The DNA of Artificial Ware). Esculapio Publishing.

Soddu, C., and E. Colabella (1995). Recreating the City's Identity with a Morphogenetic Urban Design. In *17th International Conference on Making Cities Livable,* Freiburg-im-Breisgau, Germany, Sept. 5–9, 1995.

Soddu, C., and E. Colabella (1997). Argenic Design. *The European Academy of Design, Contextual Design/Design in Context Conference,* Stockholm, April 23–25, 1997.

This Page Intentionally Left Blank

3 CHAPTER

Breeding Aesthetic Objects: Art and Artificial Evolution

Mitchell Whitelaw University of Technology, Sydney

3.1 INTRODUCTION

Computer science investigates, theorizes, and formalizes the operation of the technology of our time. It also pursues the applications of that technology: the ways in which it is used and might be used. The products of computer science make their way in the world and are applied and misapplied according to dynamics of individual whim and collective desire. The Internet is one work of computer science that has made a dramatic impact on (Western) society at large, thanks to the collective interest that it has attracted. A standard for online information delivery has given rise to an entire cultural and commercial sphere that will continue to shape successive waves of technological and cultural change.

An important premise of this chapter, then, is that the objects of computer science are actively involved in the wider dynamics of Western technological culture. Here, a very particular instance of this involvement is considered: the application of techniques of artificial evolution in the creation of works of art. Upon close examination this small, somewhat obscure intersection of artificial evolution with art-making raises a variety of interesting issues. Within this intersection we can observe implementations of artificial evolution with peculiar properties: systems that reflect the creative aims and aesthetic imaginations of the artists who use and design them. We can also observe the more abstract structures that surround and inform these systems—the notions of evolution, creativity, and the aesthetic object, the biological metaphors, the ideas of creative agency—as well as the aims and desires that propel them. From the perspective of art and culture, we can ask, what kind of creation is this? How does an evolved aesthetic object operate as an artwork? In what sense is the "breeder" of such an object an artist?

An analysis such as this is more usually found in a humanities context, under the label of cultural studies or new media theory. It is presented here as a kind of field report: Artificial evolution has moved beyond the disciplinary boundaries of computer science and become involved in the melee of cultural and artistic practice. What follows is an attempt to return an account of its cultural and artistic implementations (and implications) to the scholarly community most closely involved with the invention and development of the technique.

3.2 BREEDING AESTHETIC OBJECTS

The historical lineage of artificial aesthetic evolution begins with Richard Dawkins, who devised Biomorphs, a program in which the user can guide the "evolution" of generations of graphical stick figures. Dawkins describes his experiments in "Biomorph Land" in his popular work on Darwinian evolution, *The Blind Watchmaker* (1987, p. 51–74). While the evolution of the Biomorphs is based solely on their graphical appeal, this breeder is well outside the context of art practice. Within *The Blind Watchmaker* it is used to support Dawkins's argument for the power of cumulative selection, as a digital demonstration of the capacity of the evolutionary process.

Nonetheless, Dawkins's Biomorphs were to help inspire a succession of artists to take up artificial evolution. During the late 1980s and early 1990s William Latham, in collaboration with IBM scientist Stephen Todd, created software for synthesizing, mutating, and evolving three-dimensional forms—in the artist's words "ghosts of sculptures"—that he exhibited internationally as cibachrome prints and video animations (Todd and Latham 1992). His first major exhibition of evolved work, "The Conquest of Form" (1988–89), organized by the Arnolfini gallery in Bristol, toured UK and German galleries and museums including the Natural History Museum in London and the Deutsches Museum in Munich. Despite gaining wide critical attention, Latham has declared himself "dissatisfied" with the art scene and is no longer active within it (Computer Artworks 1998). After a second touring exhibition in 1991–92, and the publication of a book on his work with Todd (1992), he cofounded software and animation company Computer Artworks in order to develop his work "in a popular form for the mass market" (Computer Artworks 1998).

Inspired in turn by the work of both Dawkins and Latham, American animator, artist, and A-Life researcher Karl Sims developed software for the evolution of two-dimensional images around 1991. Sims presented *Genetic Images,* an artwork using this software, in 1993 at the Austrian Arts Electronica festival, and in an installation at the Paris Centre Georges Pompidou the same year. Running in real time, Sims's work allowed museum visitors to act as a collective "selector" for generation after generation of evolved images. The images themselves showed that graphic objects of remarkable complexity, and some of remarkable beauty, could be generated using evolutionary techniques (see, for example, Sims 1991, 1993a).

The work of Sims and Latham continues to be influential. The years following publication of their work have seen a number of artists and computer scientists pursue the approaches their work sets out. Projects following Latham's use of procedural/iterative constructive geometry include Andrew Rowbottom's FORM software (Rowbottom 1998) and Dancer DNA by UK company Notting Hill (Notting Hill 1999). Others, including American artist Steven Rooke (see Chapter 13), veteran "algorist" Kenneth Musgrave (Musgrave 1998), Dutch A-Life researcher Peter Kleiweg (Kleiweg 1998), and American computer scientist John Mount, have followed Sims's image-based approach. Mount's "International Interactive Genetic Art" project virtualizes Sims's *Genetic Images* installation, allowing Web users to act collectively as aesthetic selectors by evaluating the images displayed (Mount 1998).

3.2.1 A Case Study—Steven Rooke

These breeders can be treated, as they are above, as technical objects with particular characteristics and as art objects (or art processes) with names and dates. They can be readily identified and described in this way. However, as suggested earlier, there are more detailed structures to be considered, and more interesting interplays of the technical and the creative. In an account of the work of one contemporary artist working with artificial evolution, some of this detail is made apparent.

In 1991, equipped with a graphics workstation, an interest in ecology, and a background in geology and graphics programming, Steven Rooke set out to create a graphical ecosystem simulation. Quickly realizing the enormity of the task, he turned instead to the artificial evolution of images. Rooke describes himself as an image breeder "in a tradition inspired by evolutionary art pioneer Karl Sims" (Rooke 1998a). Like Sims, Rooke uses a "genome" constructed from a grammar of mathematical functions that are evaluated to produce an image; variation is induced in successive generations through random variation ("mutation") or the combination of two existing genomes (sexual reproduction). However, Rooke extends Sims's breeder in several ways. First, he enlarges the grammar of mathematical functions that the genome can draw upon, adding fractal functions to the trigonometric and arithmetic operations in Sims's library. This extended grammar is clear in Rooke's images from around 1997: many of these resemble fantastic landscapes, with the familiar swirls and filigrees of Julia and Mandelbrot fractals sweeping into the distance. Images such as *In the Beginning* (Color Plate 4) and *Skaters* (Color Plate 5) are characteristic; brightly colored and extremely detailed, they are layered with complex, intersecting structures apparently receding into infinity. The fractal functions in these images are fixed elements—fractal equations in their standard forms (Rooke's "black-box" fractals). However, in recent experiments, Rooke has begun to open these fractal equations to evolutionary variation, with remarkable results. *Hyperspace Embryo* (Color Plate 6) shows such a "genetic fractal" rendered using a statistically derived representation of the iterated function.

While these technical innovations are significant in themselves, Rooke's work is also marked by more general differences in approach and intent. Sims seemed interested in offering a first-hand sense of evolutionary process with the open interactivity of *Genetic Images.* By contrast Rooke's practice is solitary and firmly focused on the resulting image. In fact the very personal nature of Rooke's work distinguishes him within this field; rather than a calm aesthetic exploration, his search is a passionate process, strongly linked to a mystical or metaphysical vision:

> Images evolve that look like places I see in my dreams, sometimes complex landscapes I can fly through, sometimes evocative forms that seem familiar, just beyond the edge of recognition . . . I have seen these shapes and places before, in dreams, in altered states, in rocks, landforms, forests, arthropod shells, galaxies, in microscopes (Rooke 1998c).

Rooke frequently describes his process in terms of familiarity and its own urgent momentum:

> I can't stop. There is something compelling about this process. It feels as though the images are trying to break out of their hyperspace into the physical world. Sometimes I'll be two or three days into a run—dozens of generations with one or two hundred individuals in the population—when Wham! there's something familiar staring back at me from out of the computer screen, demanding to be made real (*ibid.).*

Here Rooke also indicates the scale of the evolutionary process and something of his own role as patient selector, who in the quote above is rewarded—struck—by an image.

Rooke's background in geology is evident in the language he uses to describe the image evolution. Rather than start the evolutionary process from primordial "scratch" for each image, Rooke begins with "digital amber"—genomes already evolved to a certain degree of complexity and stored for future use. (Here Rooke refers to the preservation of ancient insects and their genetic material in the solidified resin that forms amber.) This involves a reduction in the diversity of possible images that the process can access, but conversely limits the exploration to a smaller "region" that Rooke has already selected as aesthetically interesting. He likens this honing down of genetic diversity to the evolutionary "shakeout" that reduced biological diversity after the proliferation of the Cambrian period. Metaphorically, Rooke is locating his work within a strongly progressive evolutionary flow: rather than the fast, cheap diversity of an initial evolutionary boom, he is interested in the long haul, the slow evolution of higher orders of complexity. As an earlier quote indicated, the scale of this evolutionary process—the number of "generations" that an image embodies—has its own appeal for Rooke: It projects a personal creative process into the expanses of geological time.

This evolutionary language illustrates one of the two dominant metaphors involved in Rooke's work. The other involves a space—or rather, a number of spaces. The notion of parameter space is often used in discussing the products of artificial evolution; given an artificial genome with a certain number (*n*) of variables, the potential results can be imagined as occupying an *n*-dimensional space. This spatial metaphor is used throughout the lineage of artificial aesthetic evolution—notably in Dawkins (1987) and Todd and Latham (1992). In Rooke, the static, finite notion of parameter space becomes a more expansive "image hyperspace." This change is technical as well as metaphorical: In the open, algorithmic genetic structure Rooke uses there is no fixed number of variables; a mutation can add or remove variables and alter the dimensionality of the genome's parameter space. In Rooke's writing, however, "image hyperspace" becomes far more than a mathematical figure. He links it to Terence McKenna's notion of an "invisible landscape," an inner, imaginary space. In a discussion of a sensor-driven interface that would allow a more intuitive process of image selection, Rooke suggests that "something like this should lead eventually to . . . a much more fluid, interactive, richer way to pull images out of [people's heads | image hyperspace]."[1]

3.3 BREEDING AND CREATION

Rooke's work begins to raise some interesting questions regarding the evolutionary process, the role of the artist, and the relation of the two. What kind of creative agency, or creative will, is at work in these breeders? In the work of artists such as Rooke and Latham, creative agency seems initially to operate in a fairly conventional way. Artificial aesthetic evolution is the "process," and the aesthetic result, in the form of a digital print or videotape, is the "product." However, further investigation reveals a more complex relation of process, agency, and "artist" status, here and in other breeders.

---

[1] Personal communication, August 11, 1998.

3.3.1 Creative Agency and the Breeding Process

Todd and Latham (1992, p. 209) discuss the way their "evolutionism" "changes the role of the artist in creating an art work." Here that role is twofold, involving "the creation of generative systems and structures" on one level, and "the selection of specific forms and animations" on the other. While the authors anticipate that these roles might be performed by different people, such that the artist's role becomes "less clear," they vigorously distance themselves from any such confusion. They declare: "For Latham's works there is no ambiguity. Latham is clearly the artist . . ." and extend this assertion, claiming "he invented the evolutionism style . . . Latham will remain creator of the style in the same way that Picasso is the father of Cubism and Haydn of the string quartet." Latham reactively stakes a claim for ownership of "evolutionism," and a very traditional form of artistic status. Todd and Latham (1992, p. 12) also introduce an analogy for this twofold artistic role that suggests another important side to the constructions of agency operating in these systems. "The artist first creates the systems of the virtual world . . . then becomes a gardener within this world he has created." The authors frequently refer to these roles simply as "artist creator" and "artist gardener."

This "artist creator" role suggests a kind of amplified creativity at work in artificial evolution. In a conventional model of the creative process, there is a straightforward causal connection between artist and artifact. As Todd and Latham explain, artists using artificial evolution also create the generative computational systems from which their aesthetic artifacts emerge. They are engaged in what we might call *metacreation:* the formation of a system with a seemingly limitless generative potential—the creation of creation itself. In Todd and Latham that extension is imagined as the creation of a "virtual world"—vast "gardens" of aesthetic potential. The parallels with Western theology are inescapable: Kevin Kelly makes the implicit explicit when he concludes an article on Sims announcing, "The artist becomes a god, creating an Eden in which surprising things will grow" (Kelly 1994a). To another reviewer Latham's work suggests "anxiety . . . in the face of a pervasive god-like omnipotent fantasy, provoked by the possession and control of powerful technical equipment" (Barter 1992). Latham's depiction of himself as the "creator" of a virtual world replete with organic form certainly suggests such an "omnipotent fantasy."

In Sims's *Genetic Images* the artist's agency is less actively constructed. Control of that process is relinquished, turned over to the work's audience. Sims stays in the "creator" role, defining the formal structures underlying an aesthetic space that others explore. The composite or collaborative agency this suggests, the same complex agency that threatens Latham's sense of his own status, is something Sims explores with more enthusiasm. He describes the installation as "an unusual collaboration between humans and machine" that "permits the creation of results that neither of the two could produce alone" (Sims 1993b, p. 404). Sims actively questions the resemblance between more conventional creativity and artificial evolution, ultimately identifying *Genetic Images* with evolution's paradoxical designer-free creativity, a creativity of "accident" rather than "good." He speculates that the work will "challenge yet another aspect of our anthropocentric tendencies" and demonstrate the "power of the evolutionary process in general—in simulation, as well as in its many forms in the world around us."

Sims's appeals to a generalized evolutionary creativity begin to dissolve the role of the individual user or selector—just as in *Genetic Images,* an individual's sequence of preferences might be absorbed by a longer-term, more collective process of evolution. The importance of the subjective process of "breeding" aesthetic objects cannot be neglected, however. As Stephen Rooke has indicated, the process begins to assume its own momentum: "I can't stop. There is something compelling about this process . . ." (1998c). The response of the normally staid Richard Dawkins is even more remarkable: "I cannot conceal from you my feeling of exultation as I first watched these exquisite creatures emerging before my eyes . . . I couldn't eat, and that night 'my' insects swarmed behind my eyelids as I tried to sleep" (1987, p. 60).

Clearly, this process engages those who use it in a remarkable way; we might speculate as to how this engagement operates. In Sims's discussion of *Genetic Images* he explains how the variation occurring in image breeding differs from the creative alterations involved in more conventional image making (Sims 1993b). Here, creative variation is "succinctly executed by the computer" in the form of a random mutation. It is this computational mutation that is the key to an understanding of the kind of creative agency at work in these breeders—one that operates through preference and selection rather than active construction. The passive position of the artist/user is particularly important: as Sims says, the computer tirelessly, "succinctly" varies the aesthetic object and can do so quickly, easily, and endlessly. Propelled at speed through generation after generation, the artist enjoys an exhilarating excess of choice, as new objects/creatures appear and are left behind. With the effortlessness of "mutation" comes an accelerated loop of change and selection that can continue indefinitely: the "creative" process is extended into an endless deferral of its object. An analogy can be made with the psychology of shopping, an activity that in affluent cultures offers not so much necessary material objects as sheer desire, the endless promise of more. Similarly the psychology of breeding aesthetic objects is caught up with the process more than the object, a process driven by a spiral of variation, desire, and selection without apparent limits. Perhaps the strength of the desiring loop that these works set up accounts for the responses of Rooke and Dawkins.

3.3.2 The Evolved Aesthetic Object

How then do the peculiarities of this creative process, and the creative agency it involves, influence the way that the products of artificial evolution are interpreted as works of art? As this rush of "accelerated evolution" feeds itself, it risks rendering its aesthetic objects meaningless. In this overflow of image material, how can one mutant claim more significance than another? When evolution is this fast, and this easy, how do we evaluate its results?

It is not only the nature of the process that begs these questions: taken at face value, the images produced by Rooke's and Sims's breeders are vivid but completely abstract digital artifacts. The formal-procedural grammar that underpins their images is independent of scale. A single image is an arbitrary, infinitely detailed window in an infinitely large coordinate plane. While a rendering of an image at a certain resolution might produce something resembling a kind of decorative abstract painting, it will have neither the materiality nor the gestural quality, the mark of human action and agency, of such a painting; like a fractal, it recedes forever within the frame and extends indefinitely outside it. These products of aesthetic selection represent a strange coupling between an "a-scalar," "a-human" generative process and a human aesthetic perception. Kelly (1994a) describes Sims's images as "mirages . . . of an alien beauty"; however, significantly the "beauty" these artists breed often involves a move away from the inherent strangeness of their generative language and toward more conventional modes of representation. One of the examples shown in Sims (1993b) is of an image that resembles a stylized face. This image seems to demonstrate an urge to find an image of the human—a kind of mirror—in these vast abstract fields. Rooke's account of his move toward more illusionistic, landscapelike images shows a similar desire to find a familiar pictorial language. Images such as *In the Beginning* (Color Plate 4) and *Skaters* (Color Plate 5) illustrate the tension between formal generative elements (the distinctive fractal curls, flattened and stretched throughout the images) and representational convention (the horizon line, the sense of illusionistic depth).

Rooke's work also shows very clearly how the metaphors attached to the breeding process inform its products. The increasingly "spatial" nature of Rooke's images resonates strongly with the other linked spaces he discusses—image hyperspace, inner imaginary space, and McKenna's "invisible landscape." In fact it seems that Rooke's images in some sense depict the very "space" he imagines them to inhabit. As well, his images operate literally as spaces, spaces in themselves. A vast, high-resolution print of the image swallows the viewer up:

> I suppose I am known as something of a fanatic about image resolution, or detail. For me, there is never enough. I want to print these images at the largest size, with the finest resolution and quality available . . . Picture seeing a large mural from a hundred feet away. As you walk closer to it, you see increasing detail—right up to the limit of your vision 12 inches away (Rooke 1998c).

The same sense is evoked as Rooke projects slides of his works onto a group of dancers: "When people enter the images, sometimes a phenomenon happens where the dancer feels like they're snapping into and out of the image. ('Am I in the image, or is it in me?')"[2] Rooke refers to this technique as "slide immersion."

It seems that as we try to ascribe significance to the evolved aesthetic object, it returns us to the evolutionary process. The biological and spatial metaphors that frame the work of Rooke and Latham come from the process itself; their artworks are perhaps best interpreted as self-referential explorations—mystical depictions of the imaginary spaces of genetic potential, in the case of Rooke, and slightly grotesque meditations on organic form, in Latham's work. In one sense Latham is unusual here in that he offers a subtext to his own work that labors against the positive, expansive sense of evolutionary potential that characterizes the work of other artists. He describes his work as "a parody of genetic engineering" and a comment on "the wanton destruction of the natural world" (Todd and Latham 1992, pp. 207–208). Elsewhere he is quoted as arguing that his work "reminds people of things that they'd rather forget" such as "viruses, cancer [and] bodily processes" (McClellan 1998).

3.4 LIMITS

Throughout this field there is a sense of excitement at the potential of aesthetic evolutionary processes. All of the artists discussed here remind us in their writing of the extent of the genetic spaces, and hyperspaces, that their systems explore. Kevin Kelly, writing on Sims's work, describes his system as accessing "a universe

---

[2] Personal communication, August 10, 1998.

. . . of all possible pictures" (1994b, p. 340). This universe supposedly contains "all shades of rose . . . the Mona Lisa, and all Mona Lisa parodies . . . the blueprints of the Pentagon," although of course the evolved images Kelly sees are "amorphous blotches, streaks, and psychedelic swirls of colour."

Kelly prompts the question, do the image hyperspaces of Sims and Rooke actually contain all possible images? Given the open-ended structure of their genetic code, this may be difficult to determine formally—in any case such a determination is outside the scope of the current investigation. However, the images themselves suggest that, so far, a fairly limited portion of this image space has been explored; no Mona Lisa has yet emerged. Rather the images retain a distinctive algorithmic aesthetic; the generative raw materials of their "genomes," the libraries of functions they employ, combine (and recombine ad infinitum), but the aesthetic quality of the images remains relatively unchanged.

Instead of a boundless genetic space, access to "all possible pictures" (or forms), it seems that these systems set out their own specific aesthetic domains. The structures of the artificial genome, its rules of mutation and reproduction, and its means of expression combine to give rise to a particular aesthetic. While this phenomenon makes the expansive evolutionary rhetoric that appears around these systems less credible, it should not be taken as an artistic failure. Rooke's work shows how the aesthetic qualities of the evolved algorithmic image interact fruitfully with the artist's own imagination. The fractal structures in Rooke's images are related to the sense of landscape that he pursues; the landscape metaphor links in turn to the figures of inner imaginary space and computational image hyperspace. Similarly the infinite detail of the algorithmic image surface feeds into this mixture of spaces, as each image becomes a boundless space in itself.

The evolved forms of William Latham's work provide another example of this phenomenon; here the genetic structure has been designed to give rise to particular phenotypic forms: twisting spirals, branching and recursive structures, tentacles, and ribs. The evolved results are limited in their variety, but have a distinctive emergent aesthetic—a combined product of the artist's design of the underlying generative structures, and the aesthetic selection guiding their evolution. The considerable success of Latham's work shows that a specialized, and in many ways limiting, genetic structure can give rise to a highly individual aesthetic.

Nonetheless, the sense of potential that the evolutionary process offers remains palpable in this field. Rooke continues to work on a new genetic structure, one based on Sims's later work on the evolution of behavior and locomotion (Sims 1994). This directed graph genome is a more dynamic structure, resembling a genetic network. Rooke suspects that it "will be capable of much richer evolution." Rooke's plan suggests another interesting limit for aesthetic evolution; in a quest for aesthetic richness and variety, the formal models involved seem to become more complex, more dynamic, and more similar to the biological structures they imitate. As aesthetic evolution becomes more powerful, and its evolved artifacts become more complex, they may also become more lifelike. At some mythical point, the artwork may attain a kind of autonomy, a kind of life: The human artistic process will have finally exceeded itself.

3.5 DRIESSENS AND VERSTAPPEN—AN ALTERNATIVE APPROACH

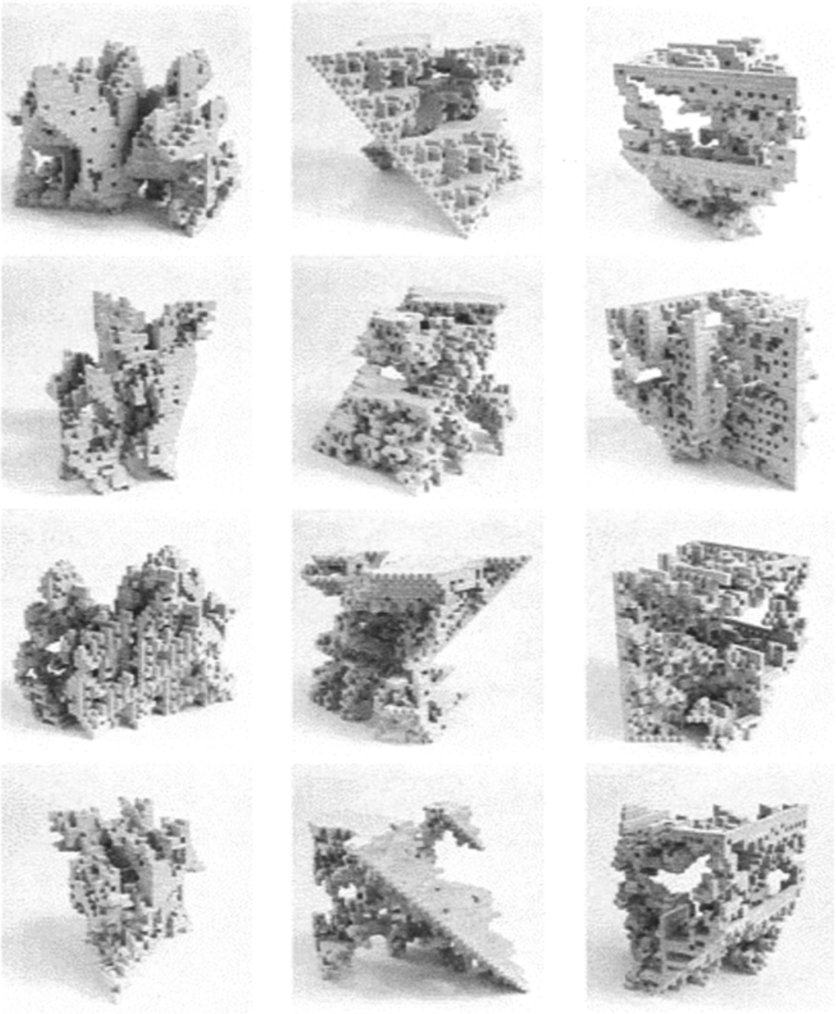
All of the works discussed above can be clearly traced to a single lineage, beginning with Dawkins's Biomorphs. As such, it is not surprising that they apply artificial evolutionary techniques in similar ways. First, they each set up a mutable artificial genotype that, when interpreted or rendered, gives rise to a single aesthetic phenotype. The biological analogy inherent in artificial evolution encourages us to understand these evolved aesthetic objects as "organisms." Second, these works all use a subjective selection process that operates as an amplified, though paradoxically "passive," version of conventional creative agency. The emphasis in these works is on the artist (or the user) choosing an evolutionary path, steering the cycle of mutation and variation according to an aesthetic preference. The implications of this approach—which is dominant in this small field—have been explored above: It leads to particular figurations of creative agency, particular questions about the nature of the evolved artifact, and more open questions about the processing limits that it seems to push against. However, as the work of Dutch artists Erwin Driessens and Maria Verstappen shows, there are alternatives to this approach.

Breed (1995–97) is a project that spans software process, evolved three-dimensional forms, and physical manifestations of those forms. Where most form breeders use additive or constructive geometry, following the work of Latham and Todd, *Breed* generates intricate three-dimensional forms through a stepwise process of spatial differentiation that the artists describe as "cellular." The process begins with a solid, cubic volume—a single cell. When the cell "divides," its volume is partitioned into eight smaller cells, units of space that may be either solid or empty; the space occupied by the initial cube is coarsely differentiated—a chunky, blocky assemblage is formed. The same process continues for each of the eight new cellular units, which divide in turn into eight, and the resultant form is again more detailed and differentiated. This process continues, being applied to rapidly increasing numbers of smaller and smaller cells. Viewed as an animated process, the initial cube carves itself away in ever-finer cellular chunks, finally resembling a complex sculpture assembled from small cubic blocks—a "pixelated" mass perforated with irregular hollows and voids (see Figure 3.1).

This morphogenetic process is controlled by a set of parameters that constitute the form's artificial genome. These parameters are in fact simple rules governing the process of differentiation—something like a three-dimensional, recursively differentiating cellular automaton. At each step in the process, a cell's subsequent differentiation is controlled by the presence (or absence) of neighboring cells; the morphogenetic rules simply dictate how every possible combination of full or empty neighbors influences the next "split." This is a highly elegant form of artificial genetics, one in which a simple, compact genome generates a complex form through recursive application at ever-finer scales, rather than through a high-level or global specification. Of course these parameters still entirely determine the resultant form—a given genome will always give rise to the same form—but the process of "expression" is tightly bound to the phenotypic context of the virtual form. The formation of a certain void or bump at a certain level of detail depends not on an explicit representation encoded in the genome, but on the genetically specified interaction of neighboring units of volume.

Unlike those already discussed, the evolutionary process in *Breed* is not user- or artist-guided, but automated as in more utilitarian applications of evolutionary computation. A form generated by a random genome is evaluated according to a given set of spatial criteria—a measurement is made of properties such as volume, surface area, and "connectivity." These values are stored, and the "genome" for that form is altered randomly. A new form is generated, and its fitness values compared with those of the initial form. If those values are higher, the new form is retained as the basis for the following "mutation"; if not, the new form is discarded and the initial form mutated again. Through repetition of this simple, automatic loop, a form will eventually emerge that has "maximal fitness," meeting those predetermined formal criteria. Interestingly, while this process gives the impression of a linear, progressive drive toward an optimal ideal, the relatively open nature of the criteria is such that they can be met by a large number of different forms. Rather than searching for a single, absolute goal, this simple stepwise evolution uses only a local comparison. As it forms a sequence of incrementally "better" forms, the process effectively paints itself into a corner: the final "optimal" form is in fact only the most optimal form that that specific sequence of random alterations had produced.

Readers familiar with evolutionary computational techniques will recognize that this stepwise evolutionary process drives the evolved forms toward local



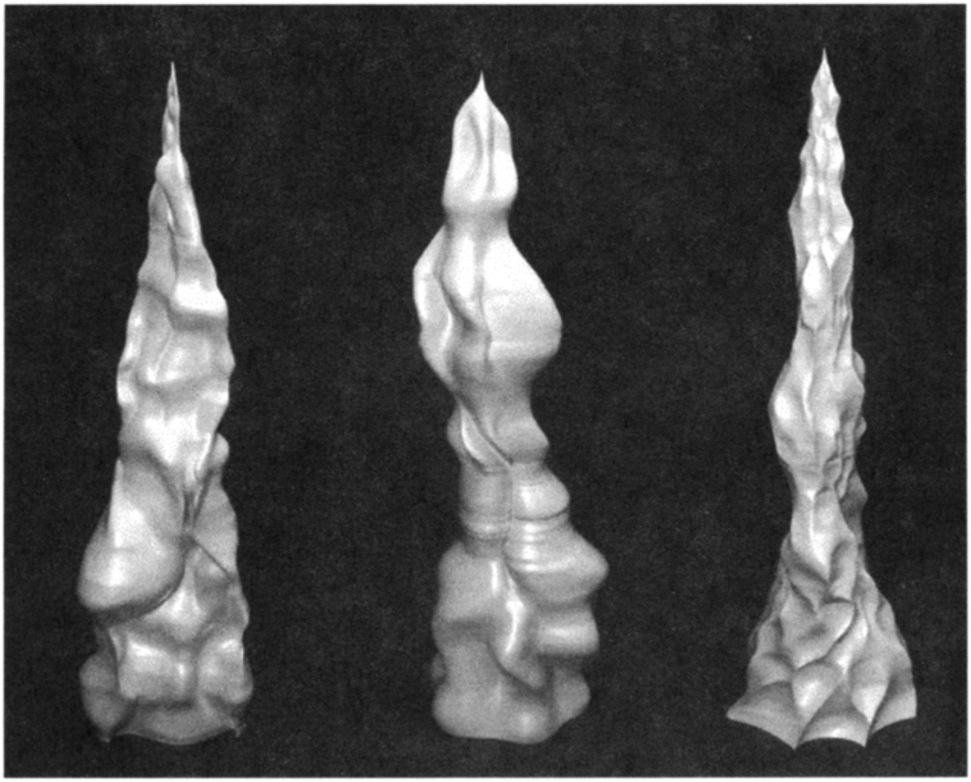
3.1 FIGURE Plywood models of forms generated using Driessens and Verstappen's *Breed* process. Each column shows four views of a single form.

maxima in a very loosely defined fitness landscape. Where utilitarian applications generally attempt to search the space of potential solutions widely, this simple process results in a self-narrowing evolutionary progression—a temporal process that in a sense matches the self-differentiating process that the forms themselves undergo. The result is a kind of loose control: Certain high-level spatial qualities are manipulable, while the low levels—the rules for cellular differentiation and their complex morphological outcomes—are left open.

Driessens and Verstappen's more recent *Tuboid* sculptures (1998–99) involve a similar use of self-constraining morphogenesis. Here, the spatial template is a tube, rather than a cube, though as in *Breed* it is a form that shapes its own development through space and time using a simple artificial evolutionary process. Here, a wormlike shape is formed from a sequence of cross-sections—two-dimensional slices—that accumulate over time. The outline of each slice is formed by joining the endpoints of a group of 32 radial, articulated "spokes"; the outline behaves like a skin stretched over this jointed skeleton. An artificial genome specifies the details of this skeleton—spoke lengths and rotations of their various segments—and these values are subject to mutation with each new slice. The results of this mutation are constrained by a simple criterion: the outline of the slice must not intersect itself; that is, it must define the perimeter of a single two-dimensional form. The result is a twisting, bulging profile that shifts through time (and its third dimension) to define a single organomorphic form. As in *Breed,* the form is not specified by the genome but rather emerges from the interaction of that genome—and in this case its mutational sequence—with its past activity and with a simple spatial constraint. Here, the form's tuboid, membrane-like quality, and the mobile virtual viewpoint, allow it to be seen as from the outside or from within—as a wormlike extrusion, or as an enfolding, unfolding tunnel, generated in real time. The artists have also shown the results of this process as hand-fabricated physical models. Figure 3.2 shows physical models of forms evolved using *Tuboid.*

These works show very clearly that artists' applications of artificial evolution need not follow the templates set out by Dawkins, Latham, and Sims. In those systems a formal genotypic representation is interpreted or solved to give a phenotype. Typically this process of "expression" is deterministic, instantaneous, and strictly one-way—a transparent act of translation. Driessens and Verstappen explore more complex processes of expression that are closely involved with the phenotypic form. Here, rather than a "blueprint," the genome is a set of rules controlling local interactions and changes. As in "wet" biology, accidents of temporal sequence are important: The *Tuboid* forms are essentially spatial records of a procession of self-constraining mutations. Similarly, the evolved volumes in *Breed* are the product of an automated self-limiting evolutionary process. These works show that matter, time, and space—constraints that other form and image breeders tend to shrug off in favor of a form of accelerated aesthetic Darwinism—can interact with generative and evolutionary processes in rich and interesting ways.

Finally, Driessens and Verstappen also show that as artists apply artificial evolution, they will begin to reshape it to serve their own ends and interests. Working at a safe distance from both the debates of theoretical biology and the



3.2 FIGURE Forms generated using Driessens and Verstappen's *Tuboid* process.

"real world" imperatives of computer science, they are ideally positioned to explore artificial evolutionary techniques. Of course, Driessens and Verstappen are not alone in these explorations. In a mutation along a different line, Steven Rooke has recently proposed a general-purpose genetic architecture that would allow for symbiogenesis—the mutually advantageous merger of individuals into a single genetic unit.[3] Inspired by the theories of biologist Lyn Margulis, such a system could entail a form of artificial evolution that overturns the conventional correlation between genotype and phenotype. As these examples show, artists can be understood not only as users of artificial evolutionary techniques, but as agents involved in the transformation of those techniques. The simple creative urge that has been so effectively answered by artificial evolutionary processes has already begun to reengineer those very processes.

---

[3] Personal communication, March 16, 1999.

3.6 CONCLUSIONS

A computational technique—artificial evolution—becomes an artistic technique. Out of this transition a set of concrete artifacts, artworks, and software emerges. However, less concrete structures emerge as well: the concepts, the metaphorical structures, and the creative aims and ambitions that surround and inform its application. Through these more abstract structures, we can observe the ways in which artificial evolution intersects with artistic practice, with the conventional categories of "work" and "artist," with notions of creativity. We can also observe some of the paradoxes at work in the field, the tensions between its expansive rhetoric and its aesthetic results.

Even as it functions, in part, within artistic practice, artificial evolution challenges it on a number of fronts. The role of the artist alters dramatically. The individual creative will is complicated by its engagement with the evolutionary process. The evolved artwork itself functions in an unconventional way. At its limit, it seems to promise an art that moves beyond human will entirely—one that may be impossible to understand as "art" at all. On the other hand, it seems that artists themselves are willing to engage with these challenges, to explore these notions while remaining ultimately in control of both process and evolved object. Further, they show how creative practice extends beyond the simple generation of aesthetically pleasing or novel objects and into the reengineering of evolutionary and morphological processes themselves.

REFERENCES

Barter, R. (1992). William Latham. *Arts Magazine* 66(1):92.

Computer Artworks (1998). The Art of William Latham. Available at *http://www.artworks.co.uk/art/art4.htm,* August 24, 1998.

Dawkins, R. (1987). *The Blind Watchmaker.* Longman Scientific and Technical.

Kelly, K. (1994a). *Genetic Images. Wired* 2.09. Available at *http://www.wired.com/wired/2.09/features/sims.html,* August 26, 1998.

Kelly, K. (1994b). *Out of Control: The New Biology of Machines.* Fourth Estate.

Kleiweg, P. (1998). Peter Kleiweg. Available at *http://odur.let.rug.nl/~kleiweg/,* October 15, 1998.

McClellan, J. (1998). William Latham. *http://www.nemeton.com/axis-mutatis/latham.html.*

Mount, J. (1998). John Mount's International Interactive Genetic Art II. *http://www.geneticart.org/cgi-bin/mjwgenformII.*

Musgrave, K. (1998). Genetic Programming, Genetic Art. Available at *http://www.seas.gwu.edu/faculty/musgrave/mutatis.html.*

Notting Hill Publishing (1999). Dancer DNA. *http://www.dancerdna.com/.*

Rooke, S. (1998a). Genetic Art Process. *http://www.aztar.net/~srooke/process.html.*

Rooke, S. (1998b). Online Portfolio. *http://www.concentric.net/~srooke/portfolio.html.*

Rooke, S. (1998c). Biographical Information. *http://www.concentric.net/~srooke/biog.html.*

Rowbottom, A. (1998). Organic, Genetic and Evolutionary Art. *http://www.netlink.co.uk/users/snaffle/form/evolutio.html.*

Sims, K. (1991). Artificial Evolution for Computer Graphics. *Computer Graphics* 25(4):325–327.

Sims, K. (1993a). Interactive Evolution. In K. Gerbel and P. Weibel (eds.), *Ars Electronica 93,* Ars Electronica.

Sims, K. (1993b). *Genetic Images.* In K. Gerbel and P. Weibel (eds.), *Ars Electronica 93,* Ars Electronica.

Sims, K. (1994) Evolving 3D Morphology and Behaviour by Competition. In R. Brooks amd P. Maes (eds.), *Artificial Life IV,* MIT Press.

Todd, S., and W. Latham (1992). *Evolutionary Art and Computers.* Academic Press.

This Page Intentionally Left Blank

CHAPTER

4 The Beer Can Theory of Creativity

Liane Gabora Vrije Universiteit, Brussels

4.1 INTRODUCTION

I had to laugh this morning while reading the *Ottawa Citizen* when I saw a crafty yet bafflingly incompetent vandal described as "He's got a full six pack, but the plastic thingy that holds them together is missing." It's a spin-off of the saying "He's one can short of a full six pack," which itself is a Canadianized version of "He's lost a few marbles" or "He's not playing with a full deck."

The newspaper description beautifully exemplifies one of the main issues of this chapter—that the interplay of variation and continuity as a creative insight is adapted from one context or circumstance to another. But also, content-wise, it's a pithy summary of another, related issue dealt with here. In order to *adapt* the idea to a new context, in order to *evolve* it in new directions, it must originally have been stored in memory in a way that implicitly identifies its relationships to *other* ideas. In other words, when it comes to creativity, how your "beer cans" are connected together is as important as how many of them there are.

This chapter explores the cognitive mechanisms underlying the emergence and evolution of cultural novelty. Section 4.2 summarizes the rationale for viewing the process by which the fruits of the mind take shape as they spread from one individual to another as a form of *evolution* and briefly discusses a computer model of this process. Section 4.3 presents theoretical and empirical evidence that the sudden proliferation of human culture approximately two million years ago began with the capacity for creativity—that is, the ability to generate novelty strategically and contextually. Finally, Section 4.4 wraps things up with a few speculative thoughts about the overall unfolding of this evolutionary process.

4.2 CULTURE AS AN EVOLUTIONARY PROCESS

When asked in an interview "What is creativity?" Russian dancer Rudolf Nureyev gave the following answer:

> It is something born from within you. It's as though you felt a need to do something, to say something, to utter, and you cannot live without uttering this sentence or writing this piece of music. It just begs to manifest itself. It is a need to express yourself first, and then to rationalize this expression. It is irrational first, rational after. I am sure Einstein had an inkling about something unknown and then came to his theory of light. And I am sure everybody has had this impulse, very much akin to sex, sexual drive, or sexual appetite, if you wish (LeMay 1990).

It is fascinating that people frequently describe their creative process through metaphorical allusion to how novelty arises in the biological world. Is this merely tactical? For example, since everyone has experienced sexual drive, does comparing the creative impulse to it lull us into a romanticized view of the extent to which the creative impulse possesses one who gives their life over to it, not voluntarily but because that is their nature? And when artists use the word "birth" to describe the process by which something vaguely intuited, and perhaps seemingly irrational, is nurtured into the rational realm of human events and understandings, are they manipulating us into viewing their works with the reverence with which we view a newborn child? Or is there really a deep isomorphism between these two processes?

4.2.1 Variation and Convergence in Biology and Culture

Increasingly, culture is being viewed as a form of evolution (for example, Campbell 1974, 1987; Cavalli-Sforza and Feldman 1981; Csikzentmihalyi 1993, 1999; Gabora 1996, 1997; Popper 1963). To see why, we need to take a closer look at what evolution is: a process wherein a stream of information incrementally adapts to environmental constraints. It requires

- a *pattern of information* (an entity that occupies a state within a space of possible states)
- a means of *varying* the pattern (exploring or transforming the space)

- a rationale for selecting or *converging* on variations that are adapted, that is, that tend to give better performance than their predecessors in the context of some problem or set of constraints (a fitness landscape applied to the space)

With each iteration of variation and selection, the evolving entity is often better able to meet the challenges and capitalize on the opportunities presented by its circumstances; in a sense it becomes a sharper "mirror image" of its environment. The situation is complicated by the fact that the entity's environment often consists largely of other entities that are *themselves* evolving, so predator-prey relationships, symbiotic alliances, and coevolutionary arms races become established.

In biological evolution, the evolving patterns of information are genes encoded as sequences of nucleotides. *Variation* arises through mutation and recombination, and natural selection weeds out those that are maladapted. In cultural evolution, the evolving patterns of information are concepts, ideas, attitudes, values, and so on.[1] Variation is generated by "creatively" combining, transforming, and restructuring them. Factors promoting convergence include biological drives, goals, desires, values, aesthetic preferences, and the associative organization of memory, which constrain how one concept evokes (in a sense, selects) another. Actually, in the cultural domain, variation and convergence tend to go hand in hand; this process will be explored in detail shortly.

The cultural analog of the *genotype* is the network of concepts and ideas that together constitute a model of reality, or worldview. The analog of *phenotype* is the way they get implemented or communicated, through facial expressions, gestures, actions, or vocalizations. Implementation incorporates syntactic features characteristic of the channel through which it is conveyed (Brooks 1986). Thus, for example, a dance step looks different with each individual who performs it. Whereas biological information gets passed on to offspring as a complete set of genes, cultural information spreads one idea at a time.

The assimilation of new ideas into a society alters the selective pressures and constraints it exerts on the individuals embedded in it, which in turn alters the generation and proliferation of future ideas. Thus culture, like nature, comprises a self-sustained system for the exploration and transformation of a space of possible patterns.

---

[1] Sometimes units of cultural information are referred to as "memes," a term I now tend to avoid because of its reductionist overtones.

4.2.2 Is More Than One Mind Necessary for Ideas to Evolve?

Sometimes a fourth item is included in the list of requirements for evolution: a way of replicating (or amplifying, as molecular biologists refer to it) the selected variations. Ideas are said to replicate through social processes such as teaching and imitation. Clearly replication is an integral component of biological evolution, and the transmission of ideas through social processes such as imitation and teaching is likewise important. But is it *indispensable?* If, for example, you were the only human left on the planet, but you could live forever, would cultural evolution grind to a halt? If you found an ingenious way to scale a mountain, you would still have come up with something new, something more adapted to the environment, something that you might go on to modify and perfect—to evolve. Your novel mountain-scaling approach might even lead you, through metaphorical analogy, to new ideas about how to elevate your mood, or expose you to mountain plants and thereby lead to novel ways of cooking them, weaving them, and so on. Thus, strictly speaking, there need not necessarily be more than one individual for culture to evolve. Evolution can occur through variation and convergence acting on a single stream of information (such as a train of thought) without anything being explicitly replicated. There is no more reason to refer to this as "replication" than to refer to your cat *now* as a "replicant" of the cat that jumped on the windowsill a few minutes ago. The conscious experience of an individual can be viewed as an evolutionary process in which variation and convergence are not separated spatiotemporally but intimately intertwined, one pattern of qualia fluidly transmuting into the next.

Nevertheless, the "culture" of a single individual would be impoverished, to say the least. Cultural variety increases exponentially as a function of the number of creative, interacting individuals. As a simple example, a single individual who invents 10 new words is stuck with just those 10. A society of 10 interacting individuals, only one of whom is creative, is no better off; there are still just 10 words. In a society of 10 nonsocial individuals, each of whom invents 10 words but does not share them, each individual still has only 10 words. But in a society where the 10 individuals invent 10 words and share them all, everyone ends up with 100 words.

The bottom line is, culture as we know it, with its explosive array of meaningful gestures, languages, and artifacts, requires the kind of parallel processing that social interaction provides. But—if you accept the views described here—it is possible for a single individual to evolve cultural novelty.

4.2.3 Meme and Variations: A Computer Model of Cultural Evolution

This can be seen clearly in Meme and Variations (MAV), a computer model of cultural evolution (Gabora 1995). The program consists of an artificial society of interacting neural-network-based agents that do not have genomes, and neither die nor have offspring, but that invent, implement, and imitate ideas, or memes. Every iteration, each agent has the opportunity to acquire a new meme, either through (1) *innovation,* by mutating a previously learned meme, or (2) *imitation,* by copying a meme implemented by a neighbor. Whereas memes do not evolve at all in the absence of innovation, meme evolution *does* take place in the absence of imitation (albeit more slowly). In other words, when agents can generate novelty, but completely ignore one another, memes still evolve.

MAV displays many phenomena found in biological evolution, such as drift,[2] and slower increase in fitness at epistatic[3] loci. It also displays phenomena unique to culture. For example, *mental simulation* (ability to assess the relative fitness of a meme before actually implementing it) and *strategic innovation* (using past experience to bias how memes mutate, as opposed to mutating at random) both increase the rate at which fitter memes evolve.

The higher the ratio of innovation to imitation, the greater the meme diversity, and the higher the fitness of the fittest meme. However, meme fitness increases most rapidly for the society as a whole with an innovation to imitation ratio of 2:1 (but diversity is then compromised). Interestingly, for the agent with the fittest memes, the less it imitates (i.e., the more computational effort reserved for its own creative efforts), the better it performs.

The approach taken in MAV can be compared with Sims's (1991) and Todd and Latham's (1992) computer models of the evolution of creativity, which, though they explore a *cultural* process, use a genetic algorithm—a model of *biological* evolution. However, although MAV is modeled after cultural evolution, it is too simple to explore many cultural phenomena. The space of possible memes is fixed and small, and (unlike real life) the fitness function that determines what constitutes a good meme is predetermined and never changes. Also, imitation and innovation are probably not as discrete as the model suggests. Nevertheless, it demonstrates the feasibility of computationally modeling the processes by which creative ideas spread through a society, giving rise to observable patterns of cultural diversity.

---

[2] The term *drift* refers to random statistical bias due to sampling error (Cavalli-Sforza and Feldman 1981).

[3] In epistasis, the fitness of an allele at one locus depends upon which allele is present at another locus.

4.2.4 Breadth-First versus Depth-First Exploration

The following distinction can be useful for gaining insight into why, as we saw in Sections 4.2.2 and 4.2.3, it isn't essential that individuals imitate (or that there even be more than one of them!) for culture to evolve. Biological creativity is largely random. Billions of mutant gametes are ejected into the world, and most are unsuccessful, but the occasional one is better than the average, and over generations it tends to increase in the population. This kind of approach—search the entire space of possibilities without devoting much effort to any one possibility, and probably at least one of them will be better than what exists now—is referred to in computer science as *breadth-first.*

Human creativity, on the other hand, is highly nonrandom. Say, for example, you are faced with the problem of how to stop a leaky faucet from dripping. If you were to try to reach a solution by mutating each feature of the dripping faucet one by one, or by recombining it with every concept from hot dogs to postmodern deconstruction, you wouldn't get very far. We generate novelty *strategically,* using an internal model of the relationships among the various elements of the problem domain, and *contextually,* responding to the specifics of how the present situation differs from previously encountered ones. This kind of approach—explore few possibilities, but choose them wisely and explore them well—is referred to as *depth-first.*

In a sense, culture embodies the best of both worlds; that is, each individual's depth-first stream of thought is embedded in a highly parallel, relatively breadth-first social matrix that provides a second, outer tier of convergent pressure. For depth-first search to be successful, of course, requires some knowledge of the topology of the fitness landscape; in other words, some "smarts" about what sort of variation and convergence would probably be beneficial. Of course, even a relatively breadth-first algorithm like biological evolution operates with a certain degree of smarts. In fact it is sometimes argued that it too is highly nonrandom, though clearly it is not strategic and contextual in the way a stream of thought is. So the distinction is just a matter of degree.

4.2.5 Dampening Arbitrary Associations and Forging Meaningful Ones

Whether or not an algorithm is breadth-first or depth-first, the notions of linkage equilibrium and disequilibrium are useful conceptual devices for gaining an overview of what is taking place during an evolutionary process. The closer together two genes are on a chromosome, the greater the degree to which they are *linked*. *Linkage equilibrium* is defined as random association among alleles of linked genes. Consider the following simple example:

> A and a are equally common alleles of Gene 1.
> B and b are equally common alleles of Gene 2.
> Genes 1 and 2 are linked (nearby on the same chromosome).

There are four possible combinations of genes 1 and 2: AB, Ab, aB, and ab. If these occur with equal frequency, the system is in a state of linkage equilibrium. If not, it is in a state of linkage *dis*equilibrium. Disequilibrium starts out high, but tends to decrease over time because mutation and recombination break down arbitrary associations between pairs of linked alleles. However, at loci where this *does not* happen, one can infer that some combinations are fitter, or more adapted to the constraints of the environment, than others. Thus when disequilibrium does not go away, it reflects some structure, regularity, or pattern in the world.

What does this have to do with creativity? Like genes, features of memories and concepts are connected through arbitrary associations as well as meaningful ones. We often have difficulty applying an idea or problem-solving technique to situations other than the one in which it was originally encountered, and conversely, exposure to one problem-solving technique interferes with the ability to solve a problem using another technique (Luchins 1942). This phenomenon, referred to as *mental set*, plays a role in cultural evolution analogous to that of linkage in biological evolution. To incorporate more subtlety into the way we carve up reality, we must first melt away arbitrary linkages among the discernable features of memories and concepts, thereby increasing the degree of equilibrium. This needn't be a particularly intellectual process; for example, the *feeling* of a particularly painful or joyous experience could be extricated from the specifics of that experience, and remanifest itself as, say, a piece of music. As we destroy patterns of association that exist because of the historical contingencies of a particular domain, we pave the way for the forging of associations that reflect genuine structure in the world of human experience that may manifest in several or perhaps all domains.

Nureyev revealed an at least intuitive grasp of this when, further on in the interview discussed near the beginning of this section, he was asked, "Has going from [dance to acting] given you any insight into the previous kind of performance styles, and into the different shades of creativity"? He responded: "If you know one subject very well, then you have the key to every other subject." If this is true, it doesn't seem particularly fair. Why should some people hold, not just the key to the discipline they have specialized in, but *all* the keys? However, looking around, there appears to be increasing evidence in support of Nureyev's claim. Complexity theory is being applied to everything from earthquake prediction, to neuroscience, to the design of an algorithm for animating the behavior of a flock of bats in *Batman*. Genetic algorithms are being used to compose music, and mathematicians are turning into fractal artists. It isn't obvious what doors will be opened by any particular key!

To briefly sum up thus far, we have examined the rationale behind viewing culture as a form of evolution—a process by which information incrementally adapts to environmental constraint through variation and convergence. In some ways these two forms of evolution operate quite *differently;* for example, cultural novelty is generated in a more depth-first manner. In other respects they are similar, and features of biological evolution—such as drift and linkage equilibrium—transfer readily to culture. We now turn from the general principles underlying these evolutionary systems to the question of how they began in the first place.

4.3 CREATIVITY AS THE ORIGIN OF CULTURE

The origin of culture is sometimes unquestioningly equated with the onset of the capacity for social learning, and particularly imitation (for example, Blackmore 1999). But the idea that imitation is what makes us human seems counterintuitive. When we feel proud to belong to the human race, we think of the Great Pyramids, beautiful music, the airplane . . . in short, the fruits of *creativity.* The word "imitation" is, in fact, often used to denote inferiority. Are our intuitions about what makes us special actually misguided? I think not. There are several sources of evidence that creativity, not imitation, is what makes us human.

4.3.1 Theoretical Evidence

We saw in the MAV computer model that when the agents' ability to imitate is set to maximum, and their ability to invent is turned off, nothing happens. This makes sense. There has to be something worth imitating before the ability to imitate comes in handy, or even manifests itself. A society of individuals who can imitate, but not invent, is stagnant. Therefore, the suggestion that it was the appearance of imitation that brought about the onset of culture is not theoretically sound.

On the other hand, recall how when the MAV agents' ability to *invent* is set to maximum, and their ability to *imitate* is turned off, evolution does take place, albeit more slowly than with imitation. Since, as we have seen, cultural evolution is more depth-first than biological evolution, once the capacity for creativity presents itself, culture can evolve, whether or not there is imitation. Novelty can then breed more novelty. Or as one choreographer put it: "If we don't do what our predecessors did, we're doing what our predecessors did." Thus the proposal that culture originated with the onset of creativity is at least theoretically possible.

4.3.2 Archeological Evidence

Human culture is generally thought to have originated somewhere around 1.7 million years ago, during the time of *Homo erectus*. This period marks the appearance of sophisticated stone tools and habitats, use of fire, long-distance hunting strategies, and migration out of Africa, as well as a rapid increase in brain size (Bickerton 1990; Corballis 1991). This increase in variety of artifacts and habitats is exactly what one would expect to see if humans suddenly acquired the capacity to be creative. It is the *opposite* of what one would expect if they suddenly acquired the ability to imitate. If it were imitative capacity that had suddenly arisen, then just prior to this time there would have been a great variety of tools, habitats, and so on, and the onset of imitation would have funneled this variation in just a few of the most useful directions. Thus the archeological evidence is consistent with the thesis that creativity, not imitation, was the funnel for culture.

4.3.3 Evidence from Animal Behavior

Experimental research indicates that imitation (and related phenomena such as emulation, response facilitation, etc.) is widespread in the animal kingdom. It has been documented in budgerigars (Galef et al. 1986), quail (Akins and Zentall 1998), cowbirds (King and West 1989), rats (Heyes and Dawson 1990; Heyes, Dawson, and Nokes 1992), monkeys (Beck 1976; Hauser 1988; Nishida 1986; Westergaard 1988), orangutans (Russon and Galdikas 1993) and chimpanzees (Goodall 1986; Mignault 1965; Sumita, Kitaham-Frisch, and Norikoshi 1985; Terrace et al. 1979; Whiten 1998). Nevertheless, as many authors have pointed out, although imitation is commonplace, no other species has anything remotely approaching the complexity of human culture (for example, Darwin 1871; Plotkin 1988).

As we have seen, imitative capacity remains latent, hidden from view, until there is variation for it to work on. Thus the lack of cultural complexity in animals despite evidence that, when put to the test, they can imitate, is consistent with the "creativity as funnel for culture" proposal.

4.4 WHAT CAUSED THE ONSET OF CREATIVITY?

What then could have caused humans to suddenly be capable of generating strategic, contextual novelty? For a stream of creative thought to unfold, related memories and sensorimotor behaviors (some hard-wired, some learned) must become woven into an interconnected conceptual web, or worldview. However, this presents the following paradox. Until a mind incorporates *relationships* between memories, how can one thought evoke another? And until one thought can evoke another, how are relationships established among memories so that they become an interconnected worldview?

The origin of life presents an analogous paradox: If living organisms come into existence when other living things give birth to them, how did the first organism arise? That is, how did something able to reproduce itself come to be? By combining an insight from random graph theory with the concept of hypercycles Kauffman (1993) arrived at the hypothesis that life may have begun with, not a single molecule capable of replicating *itself*, but an *autocatalyticaly closed* set of *collectively* self-replicating molecules. (Note that it is *not* closed in the sense that new molecules cannot be incorporated into the set. It is closed in a mathematical sense, but not a physical sense.)

An analogous line of reasoning can be applied to explain how discrete memories become woven into a worldview. Although this account focuses on integration of the worldview through the emergence of deeper, more general concepts, the principles apply equally to integration of the psyche through the purification of intentions and emotions. A detailed account of the proposal can be found in Gabora (1998), and elaborations in Gabora (1999, 2000), but the basic line of reasoning goes as follows. Much as catalysis increases the number of different polymers, which in turn increases the frequency of catalysis, reminding events increase concept density by triggering *abstraction*—the formation of abstract concepts or categories such as "tree" or "big"—which in turn increases the frequency of remindings. And just as catalytic polymers reach a critical density where some subset of them undergoes a phase transition to a state where there is a catalytic pathway to each polymer present, concepts reach a critical density where some subset of them undergoes a phase transition to a state where each one is retrievable through a pathway of remindings events or associations. Finally, much as autocatalytic closure transforms a set of molecules into an interconnected and unified living system, conceptual closure transforms a set of memories into an interconnected and unified worldview. Memories are now *related* to one another through a network of abstract concepts; the more abstract the concept, the greater the number of other concepts that fall within a given distance of it in conceptual space and therefore are potentially evoked by it. For instance, your concept of "depth" is deeply woven throughout the matrix of concepts that constitute your worldview; it is latent in experiences as dissimilar as "deep swimming pool," "deep-fried zucchini," and "deeply moving book."

It seems likely that to produce a stream of meaningfully related yet potentially creative remindings, memories need to be distributed (though their distribution must be constrained). The process described here would most likely have been kickstarted by a genetic mutation leading to decreased neuron activation threshold, causing the storage and retrieval of memories to become more widely distributed.

So, to return to the Bob and Doug MacKenzie parlance with which this chapter began, culture may have begun with the emergence of a "plastic thingy"—a hierarchical network of abstract concepts that connects associated memories into a worldview. Some experiences are either so consistent, or so inconsistent, with this worldview that they have little impact on it. Others percolate deep, renewing our understanding of myriad other concepts or events. The worldview is stable if it fosters thought trajectories that enhance individual well-being. Much as biological organisms assimilate the food necessary for maintenance and growth but shield off toxins, an individual assimilates stimuli that expand its worldview, but censors or represses stimuli or memories that could bring harm.

A relationally and hierarchically structured worldview would be invaluable in biasing the generation of novelty in directions that are likely to be fruitful. (How this could work is discussed in depth in the extended version of this chapter on the CD-ROM.) Actually, a relationally and hierarchically structured worldview probably aids imitation too, especially the imitation of actions, vocalizations, or artifacts that are particularly complex (Byrne and Russon 1998). For example, a novel mannerism or expression could be generated by putting your own slant on it on the fly, *as* you imitate. So the dichotomy between creativity and imitation may be less polarized than the preceding discussion implies. Nevertheless, imitation *without* creativity is not sufficient to bring about cultural evolution.

4.5 CONCLUSIONS

It may be that this chapter has barely touched on what lies at the crux of creativity. It may turn out that religious and spiritual traditions have more to offer here of explanatory value than does science. Nevertheless, the notion that cultural, as well as biological, information *evolves* through variation and convergence surely gives new vitality to our understanding of the creative process, both at the individual level and the cultural.

Of course, there are significant differences between biological and cultural novelty. In the biological realm, novelty is generated by combining and mutating information units. But in the cultural realm, all sensory information gets more or less thrown into one big melting pot (or keg, you might say), that is, the conceptual network, or worldview, and novelty is generated *strategically and contextually,* by highlighting those aspects of the worldview most relevant to the current situation. It may be that the origin of the capacity to weave memories into a worldview through the formation of abstractions gave rise to the origin of culture. And you could say that creativity is the crystallized precipitate of these worldview weavings.

Everyone is creative. Every inkling in every mind alters, however minutely, the structure of a worldview that, through the spoken word, the holding of hands, and so on, interacts with other worldviews and is part of a vast chain of worldviews evolving through space and time. Nevertheless it is interesting to ask: Was Einstein a seven-pack? Or was he a regular six pack like most of us, but one in which the plastic thingy melted and resolidified in a truly exceptional way? Probably both. But the second is more likely what led him to be "person of the century" (according to *Time* magazine). The greater the number of concepts, the more levels of hierarchically structured abstraction are possible, but only *some* of these capture regularity of the world and therefore have meaning. So once there's a certain number of concepts in there, what matters most is how the individual weaves them together. In fact, up to an IQ of 120, intelligence and creativity are correlated, but past 120 they are not (Barron 1963). My guess is that Einstein had a tendency toward defocused attention and conceptual fluidity, and as a result, his worldview was less a product of what he was taught, and more a self-made construction from the bottom up. And the worldview weavings of the human race will, of course, never be the same.

ACKNOWLEDGMENTS

I would like to thank Beverlee Moore for comments on the manuscript and acknowledge the support of the Center Leo Apostel and Flanders AWI-grant Caw96/54a.

REFERENCES

Akins, C. K., and T. R. Zentall (1998). Imitation in the Japanese Quail: The Role of Reinforcement of Demonstrator Responding. *Psychonomic Bulletin and Review* 5:694.

Barron, F. (1963). *Creativity and Psychological Health.* Van Nostrand.

Beck, B. B. (1976). Tool Use by Captive Pigtailed Monkeys. *Primates* 17:301–310.

Bickerton, D. (1990). *Language and Species.* University of Chicago Press.

Blackmore, S. (1999). *The Meme Machine.* Oxford University Press.

Brooks, V. (1986). *The Neural Basis of Motor Control.* Oxford University Press.

Byrne, R. W., and A. E. Russon (1998). Learning by Imitation: A Hierarchical Approach. *Behavioral and Brain Sciences* 21(5):667–721.

Campbell, D. T. (1974). Evolutionary Epistemology. In P. A. Schlipp (ed.), *The Library of Living Philosophers, Vol. XIV: The Philosophy of Karl Popper,* Open Court.

Campbell, D. (1987). Evolutionary Epistomology. In G. Radnitzky and W. W. Bartley III (eds.), *Evolutionary Epistomology, Rationality, and the Sociology of Knowledge,* Open Court.

Cavalli-Sforza, L. L., and W. W. Feldman (1981). *Cultural Transmission and Evolution: A Quantitative Approach.* Princeton University Press.

Corballis, M. C. (1991). *The Lopsided Ape: Evolution of the Generative Mind.* Cambridge University Press.

Csikzentmihalyi, M. (1993). *The Evolving Self: A Psychology for the Third Millenium.* Harper Collins.

Csikzentmihalyi, M. (1999). Implications of a Systems Perspective for the Study of Creativity. In R. J. Sternberg (ed.), *Handbook of Creativity,* pp. 313–335.

Darwin, C. (1871). *The Descent of Man.* John Murray.

Gabora, L. (1995). Meme and Variations: A Computer Model of Cultural Evolution. In L. Nadel and D. Stein (eds.), *1993 Lectures in Complex Systems,* Addison-Wesley.

Gabora, L. (1996). A Day in the Life of a Meme. *Philosophica* 57:901–938. Invited manuscript for special issue on concepts, representations, and dynamical systems. *http://www.lycaeum.org/~sputnik/Memetics/day.life.txt.*

Gabora, L. (1997). The Origin and Evolution of Culture and Creativity. *Journal of Memetics: Evolutionary Models of Information Transmission* 1(1). *http://www.cpm.mmu.ac.uk/jom-emit/1997/vol1/gabora_l.html.*

Gabora, L. (1998). Autocatalytic Closure in a Cognitive System: A Tentative Scenario for the Origin of Culture. *Psycholoquy* 9:67. *http://www.cogsci.soton.ac.uk/cgi/psyc/newpsy.*

Gabora, L. (1999). Weaving, Bending, Patching, Mending the Fabric of Reality: A Cognitive Science Perspective on Worldview Inconsistency. *Foundations of Science* 3(2):395–428.

Gabora, L. (2000). Conceptual Closure: Weaving Memories into an Interconnected Worldview. In G. Van de Vijver and J. Chandler (eds.), *Closure: Emergent Organizations and Their Dynamics,* Annals of the New York Academy of Sciences, vol. 901.

Gabora, L. (Submitted). Amplifying Phenomenal Information: From Double Aspect Theory to Human Consciousness.

Galef, B. G., L. A. Manzig, and R. M. Field (1986). Imitation Learning in Budgerigars: Dawson and Foss (1965) Revisited. *Behavioural Processes* 13:191–202.

Goodall, J. (1986). *The Chimpanzees of Gombe: Patterns of Behavior.* The Belknap Press.

Hauser, M. D. (1988). Invention and Social Transmission: New Data from Wild Vervet Monkeys. In R. Byrne & A. Whiten (eds.), *Machiavellian Intelligence,* Clarendon Press.

Heyes, C. M., and G. R. Dawson (1990). A Demonstration of Observational Learning in Rats Using a Bidirectional Control. *Quarterly Journal of Experimental Psychology* 42B:59–71.

Heyes, C. M., G. R. Dawson, and T. Nokes (1992). Imitation in Rats: Initial Responding and Transfer Evidence from a Bidirectional Control Procedure. *Quarterly Journal of Experimental Psychology. Section B: Comparative and Physiological Psychology* 45B:229–240.

Kauffman, S. A. (1993). *Origins of Order.* Oxford University Press.

King, A. P., and M. J. West (1989). Presence of Female Cowbirds *(Molothrus Aterater)* Affects Vocal Imitation and Improvisation in Males. *Journal of Comparative Psychology* 103:39.

LeMay, P. (1990). Nureyev Now. *Dance* (May):32–38.

Luchins, A. S. (1942). Mechanization in Problem Solving. *Psychological Monographs* 54:248.

Mignault, C. (1965). Transition between Sensorimotor and Symbolic Activities in Nursery-Reared Chimpanzees *(Pan troglodytes). Journal of Human Evolution* 14:747–758.

Nishida, T. (1986). Local Traditions and Cultural Transmission. In B. B. Smuts, D. L. Cheney, R. M. Seyfarth, R. W. Wrangham, and T. T. Struhsaker (eds.), *Primate Societies,* University of Chicago Press.

Plotkin, H. C. (1988). *The Role of Behavior in Evolution.* MIT Press.

Popper, K. (1963). *Conjectures and Refutations.* Routledge and Kegan Paul.

Russon, A. E., and B. M. F. Galdikas (1993). Imitation in Free-Ranging Rehabilitant Orangutans. *Journal of Comparative Psychology* 107:147–161.

Sims, K. (1991). Artificial Evolution for Computer Graphics. *Computer Graphics* 25(4):319–328.

Sumita, K., J. Kitahara-Frisch, and K. Norikoshi (1985). The Acquisition of Stone-Tool Use in Captive Chimpanzees. *Primates* 26:168–181.

Terrace, H. S., L. A. Petitto, R. J. Sanders, and T. G. Bever (1979). Can an Ape Create a Sentence? *Science* 206:891–902.

Todd, S., and W. Latham (1992). *Evolutionary Art and Computers.* Academic Press.

Waal, F. (1991). Complementary Methods and Convergent Evidence in the Study of Primate Social Cognition. *Behaviour* 118:297–320.

Westergaard, G. C. (1988). Lion-tailed Macaques *(Macaca silenus)* Manufacture and Use of Tools. *Journal of Comparative Psychology* 102:152–159.

Whiten, A. (1998). Imitation of the Social Structure of Actions by Chimpanzees *(Pan troglodytes). Journal of Comparative Psychology* 112:270.

This Page Intentionally Left Blank

II PART

EVOLUTIONARY MUSIC

5 GenJam: Evolution of a Jazz Improviser
John A. Biles

6 On the Origins and Evolution of Music in Virtual Worlds
Eduardo Reck Miranda

7 Vox Populi: Evolutionary Computation for Music Evolution
Artemis Moroni, Jônatas Manzolli, Fernando Von Zuben, Ricardo Gudwin

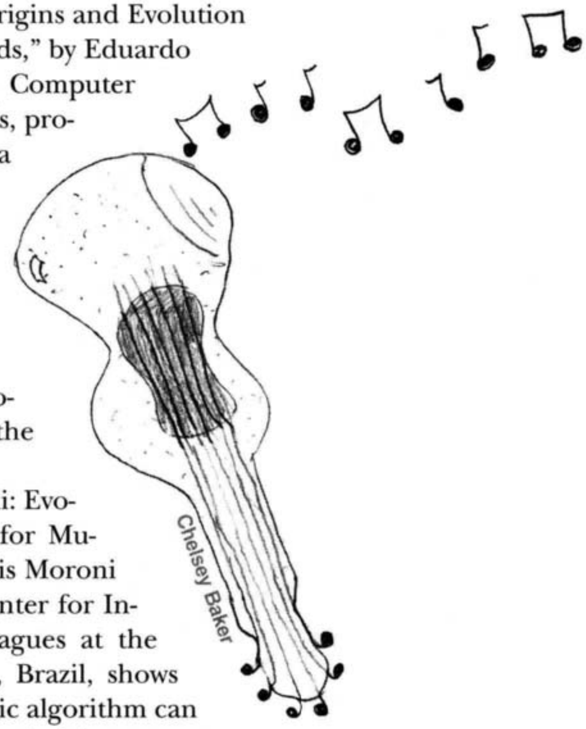
8 The Sound Gallery—An Interactive A-Life Artwork
Sam Woolf, Adrian Thompson

Music is a complicated, varied, and highly creative thing. To be able to compose or to play an instrument with feeling requires skill, imagination, and empathy. So what has evolution to do with it? Well, this part provides the answers. From contemporary music to jazz, we examine some of the most well-known and promising new evolutionary methods for the generation of "nice noises."

We begin with a chapter from John Biles of the Rochester Institute of Technology, "GenJam: Evolution of a Jazz Improviser." Here, John describes his famous software for jazz improvisation using an interactive genetic algorithm. His software is remarkable to hear: musical accompaniment being dynamically composed and played in real time as John plays his trumpet. We have also included examples for you to listen to on the CD-ROM with this book.

Chapter 6, "On the Origins and Evolution of Music in Virtual Worlds," by Eduardo Reck Miranda of Sony Computer Science Laboratory, Paris, provides a description of a very different kind of system (this time using cellular automata) that generates a very different kind of music. For those that like atmospheric and contemporary music, listen to the CD-ROM for examples.

Chapter 7, "Vox Populi: Evolutionary Computation for Music Evolution," by Artemis Moroni of the Technological Center for Informatics and his colleagues at the University of Campinas, Brazil, shows how an interactive genetic algorithm can become a new form of musical instrument, allowing the creation of new, dynamic musical structures.



The last chapter in this part, "The Sound Gallery—An Interactive A-Life Artwork," by Sam Woolf and Adrian Thompson of the University of Sussex at Brighton, investigates one of the newest approaches to the generation of music. Sam and Adrian show how evolvable hardware can be used to evolve new electronic circuits capable of making more attractive sounds based on the responses of a live, interactive audience.

5 CHAPTER

GenJam: Evolution of a Jazz Improviser

John A. Biles Rochester Institute of Technology

5.1 INTRODUCTION

At the time of writing, GenJam is about six years old, and it has come a long way from its initial conception and implementation. What began as a project intended to generate a couple of publications during a sabbatical year has evolved into a viable performance system that plays gigs two or three times a month. GenJam's current repertoire comprises over 160 tunes in a variety of jazz, Latin, and new age styles. It can perform in 4/4, 3/4, 5/4, and 12/8 time, and it has allowed the author to work as a single trumpet player in a climate that no longer supports hiring live jazz quintets. The Virtual Quintet can compete with single pianists and guitar players, something a trumpet player cannot do alone. More importantly, it allows the Virtual Quintet to compete with the dreaded DJ!

At a technical level, GenJam stretches the basic evolutionary algorithm paradigm in some interesting ways. It uses two hierarchically interrelated populations to represent its store of melodic ideas for creating improvisations. The goal of its search is to build entire populations of good melodic ideas, not just search for a single individual, which magnifies the problem of premature convergence. Consequently, many of GenJam's mutation operators promote diversity in its populations. GenJam's mutation operators are unusual in that they do considerably more than flip the occasional bit. In fact, most of GenJam's knowledge of improvisational development is embedded in its musically meaningful mutation operators. Finally, GenJam uses its evolutionary machinery for real-time interactive performance by evolving what it hears a human play into what it plays in response.

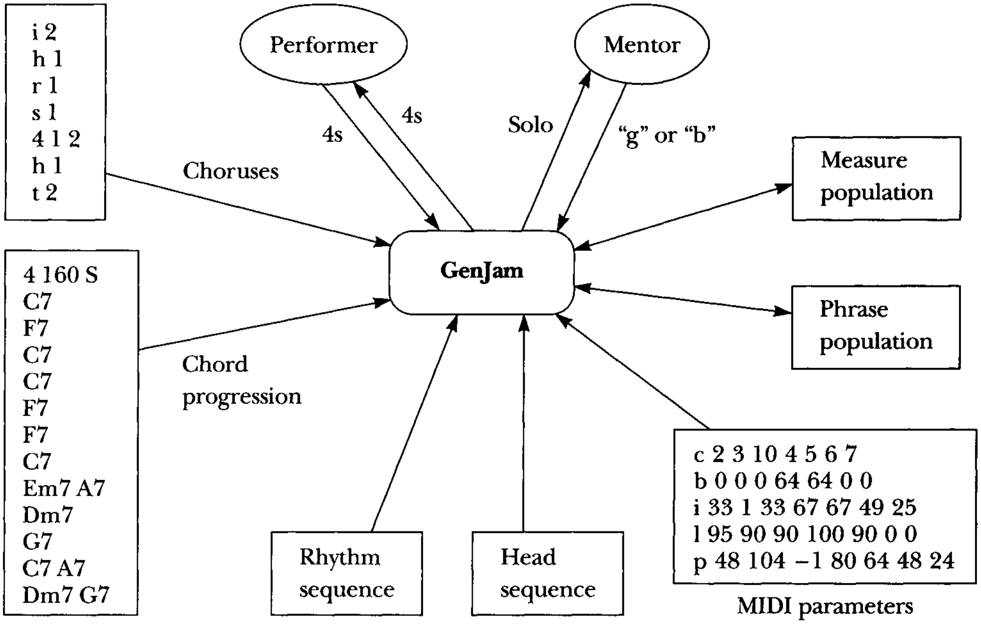
At an artistic level, GenJam's aspiration is to be received by its audience as a competent straight-ahead jazz improviser. This places GenJam in the mainstream artistically, a place most computer musicians seem to avoid. GenJam makes no attempt to extend the conception of improvisation or make any profound artistic statement on the nature of music. It simply tries to meet the audience's expectations for what straight-up jazz sounds like. Instead of challenging an audience artistically, then, GenJam tries to challenge an audience technically by achieving an accessible and convincing performance from a computer. This chapter describes how GenJam adapts the evolutionary paradigm to meet this goal.

5.2 OVERVIEW AND ARCHITECTURE

GenJam is a C program written on top of Roger Dannenberg's Carnegie Mellon MIDI Toolkit (Dannenberg 1993). The Virtual Quintet performs with an old Macintosh Powerbook 180 because of its simplicity and reliability. Other hardware includes a Yamaha MU 80 tone generator, a Roland GI-10 pitch-to-MIDI converter, and a stereo sound system. GenJam's system architecture is summarized in Figure 5.1. When GenJam plays a tune, it accesses several files and can interact with a human mentor, who trains GenJam, and a human performer, who converses with it musically. This section will describe the various files GenJam accesses to provide an overview of the information GenJam has to work with, and it will introduce the mentor and performer roles played by humans.

The rhythm sequence and head sequence are standard MIDI files that are played during the performance of the tune. The rhythm sequence provides the rhythm section accompaniment, typically piano, bass, and drum tracks, often augmented by guitar and "string" tracks. This sequence could be prepared in a variety of ways, but the author has had good luck with Band in a Box, a commercial autoaccompaniment product (Gannon 1991). The head sequence provides prescored harmony parts for GenJam to play on the "head," which is jazz-speak for the arranged melody usually performed in the first and last chorus of the tune. One liability of using "canned" accompaniment is that it will be the same in repeated performances of a given tune. However, the major focus of jazz is improvisation, which with GenJam (and hopefully with the human as well) will be spontaneous. The artistic decision was to have a rhythm section that was competent, if unspectacular, and focus on the improvisations to provide interest.

The chord progression file describes the tune to be played, including its chord changes, which serve as a harmonic road map for GenJam's improvisations. The first line of this file indicates the pitch range GenJam should use (the "4" in the example indicates a tenor sax range), the tempo (160 beats per minute in this case), and whether GenJam should use swing eighth notes (as the "S" indicates in this case) or even eighth notes, which would be appropriate for Latin tunes. Beginning with the second line of this file, each line represents a measure


i 2
h 1
r 1
s 1
4 1 2
h 1
t 2
Choruses
Performer
4s
4s
Mentor
Solo
"g" or "b"
Measure population
GenJam
Phrase population
4 160 S
C7
F7
C7
C7
F7
F7
C7
Em7 A7
Dm7
G7
C7 A7
Dm7 G7
Chord progression
Rhythm sequence
Head sequence
c 2 3 10 4 5 6 7
b 0 0 0 64 64 0 0
i 33 1 33 67 67 49 25
l 95 90 90 100 90 0 0
p 48 104 −1 80 64 48 24
MIDI parameters

5.1 FIGURE GenJam system architecture.

of the tune being played. GenJam can accept up to two chords per measure, which is a sufficient harmonic tempo for most jazz tunes. In the example, we have a standard 12-bar blues in the key of C.

The choruses file indicates what GenJam should do on each chorus of the form defined in the chord progression file. Each line represents a section of the tune. In this case, there is a two-measure introduction ("i 2"), followed by a one-chorus head ("h 1"), one chorus of rest (leaving a hole for the human to solo for a chorus), a one-chorus solo by GenJam, two choruses of trading fours between GenJam and the human soloist with GenJam going first, a one-chorus head, and finally a two-measure tag. The concept of trading fours will be explained shortly.

The MIDI parameters file provides settings for the tone generator used to render GenJam and the rhythm section, including instrument patches, loudness, stereo pan, and about 30 other parameters. One of the advantages of a virtual quintet is that GenJam and the rhythm section can play whatever instruments are available on the tone generator being used. For example, GenJam plays tenor sax, flute, pan flute, English horn, French horn, bassoon, steel drum, and various synthesizer patches. Such a versatile sideman is hard to find!

The measure and phrase population files represent a GenJam soloist. The measure population consists of individuals that decode to a measure of melodic content. The phrase population consists of individuals that decode to pointers to four individuals in the measure population. Both of these hierarchically interdependent populations are created and manipulated by GenJam during the training process, whereby a soloist is evolved under the guidance of the mentor.

The mentor is essentially the human evaluation function that indirectly provides fitness values for the individuals in both populations. This makes GenJam a classic interactive evolutionary algorithm, which is often necessary in artistic applications, due to the difficulty in developing algorithms that can determine artistic merit (Biles, Anderson, and Loggi 1996). The mentor trains GenJam by listening to it improvise on a tune and typing "g" for good and "b" for bad whenever so moved. Each "g" increments the fitness for the currently playing measure and phrase individuals, and each "b" decrements those fitness values. A heuristically derived delay shifts the allocation of feedback to accommodate the mentor's need to consider what is being played before responding (Biles 1999).

One of the most exciting traditions in jazz is the chase chorus, where soloists trade fours or eights. The rhythm section plays the form of the tune as they would for a full-chorus solo, but the soloists take turns improvising, changing off every four measures (trading fours) or every eight measures (trading eights). This typically gets competitive, with each soloist trying to top the previous one. A good tactic is to mine the previous soloist's four for a melodic idea and refine that idea in the next four. This requires good ears, a quick mind, and "nimble fingers" to pull off well, but the results can be impressive. GenJam accomplishes this oneupmanship by listening to the human performer's last four through a pitch-to-MIDI converter, mapping what it heard to the chromosome structure it uses for a phrase and four measures, possibly mutating those chromosomes, and playing them as its next four. This process, which elevates GenJam above the level of a simple music-minus-one system, will be detailed in the section on real-time interaction, but first, we must learn how GenJam represents melodic material in its chromosomes.

5.3 REPRESENTATIONS

Like all evolutionary algorithms, the choice of representation scheme is critical to the efficiency if not the efficacy of the resulting system. GenJam uses two hierarchically interrelated populations to mirror the hierarchical nature of music. These populations represent two layers in a larger hierarchy that encompasses

GenJam's conception of a jazz tune. Briefly, a tune consists of a sequence of choruses; a chorus is a sequence of phrases; a phrase is a sequence of measures; and a measure is a sequence of events. The IGA operates on the phrase and measure levels, evolving populations for each level in parallel.

The measure population consists of 64 individuals, each of which decodes to a measure of eighth-note length events. Each event is represented by four bits, which means that there are 16 distinct events possible. It also means that in 4/4 time, a measure chromosome consists of 32 bits. There are three classes of events—rests, holds, and new notes. Rest events (coded as 0) are performed as a MIDI note-off event. Hold events (coded as 15) are performed by doing nothing, which results in the prior event being held through the hold event's time window. New-note events (coded as 1–14) are performed as a MIDI note-off followed by a MIDI note-on, using the event value as an offset into roughly two octaves of the scale suggested by the chord being played by the rhythm section at the time the event is performed. These chords came from the chord progression file.

Table 5.1 lists the 17 families of chords GenJam recognizes, along with the scales it uses for each chord type, using C as an example root tone. These scales are used to build a 14-element array of pitches in the range selected for GenJam's performance in the chord progression file. In other words, the scales are normalized to an instrument's range so that there is a smooth transition of available pitches from one chord to the next. Jazz theory buffs will notice several hexatonic (six-note) scales that avoid certain notes. This was done to insure that GenJam played "safe" notes that will sound appropriate regardless of the harmonic function a chord serves. These scales represent something of a consensus of several jazz theory treatises (Russell 1959; Coker 1964; Haerle 1980, 1989; Sabatella 1992; Levine 1995), augmented by the author's experience in working with GenJam.

An important feature of the measure representation is that it integrates pitch and rhythmic structures. One problem with many music representation schemes is that pitch and rhythm are represented separately, which leads to problems when the representations must be synchronized into a coherent whole. With GenJam the rest and hold events provide rhythm at the cost of about half a bit per event. Another important feature of this representation is the fact that any string of 32 bits will decode into a playable measure. Furthermore, since the new-note events always map to pitches appropriate to the chords being played, GenJam cannot play a theoretically wrong note. This is critical to GenJam's ability to trade fours using an error-prone pitch-to-MIDI converter, as we shall see.

On the negative side, GenJam is restricted rhythmically to eighth-note multiples. At medium and fast tempos (150 beats per minute and above) this is not a

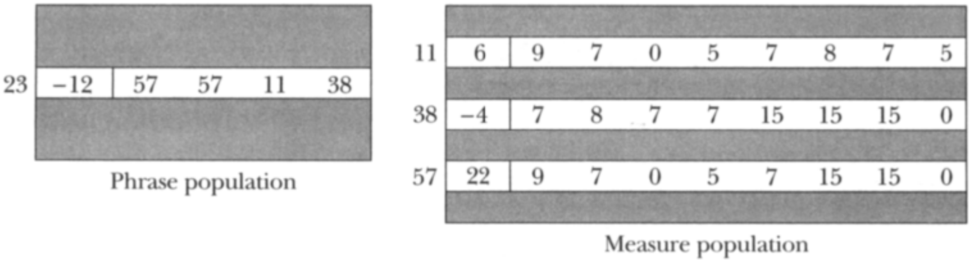
Chord	Scale	Notes
Cmaj7	Major (avoid 4th)	C D E G A B
C7	Mixolydian (avoid 4th)	C D E G A B♭
Cm7	Minor (avoid 6th)	C D E♭ F G B♭
Cm7♭5	Locrian (avoid 2nd)	C E♭ F G♭ A♭ B♭
Cdim	W/H Diminished	C D E♭ F G♭ G# A B
C+	Lydian Augmented	C D E F# G# A B
C7+	Whole Tone	C D E F# G# B♭
C7#11	Lydian Dominant	C D E F# G A B♭
C7alt	Altered Scale	C D♭ D# E G♭ G# B♭
C7#9	Mix. #2 (avoid 4th)	C E♭ E G A B♭
C7♭9	Harm Minor V (no 6th)	C D♭ E F G B♭
CmMaj7	Melodic Minor	C D E♭ F G A B
Cm6	Dorian (avoid 7th)	C D E♭ F G A
Cm7♭9	Melodic Minor II	C D♭ E♭ F G A B♭
Cmaj7#11	Lydian	C D E F# G A B
C7sus	Mixolydian	C D E F G A B♭
Cmaj7sus	Major	C D E F G A B

5.1 TABLE Chord-scale mappings.

problem, but at slow tempi (below 120 beats per minute), GenJam can sound less than convincing. The downside to GenJam's inability to play a "wrong" note is that it can't break harmonic rules in interesting ways the way good human soloists do. This reflects a fundamental design decision, opting for GenJam to play competently rather than brilliantly. Breaking the rules can sound brilliant in the appropriate context, but without a great deal of experience and taste, it usually sounds incompetent.

The phrase population consists of 48 individuals, each of which decodes to four pointers to measures in the measure population. In other words, GenJam uses four-measure phrases. Since the measure population is 64 individuals, which is 2^6, a measure pointer is six bits, which means that the chromosome for phrase individuals is 24 bits. As long as the measure population is a power of two, the phrase chromosome will be robust in that any combination of bits will map to a legitimate four-measure phrase; 64 was chosen as the measure population size over 32 or 128 because it provides enough diversity without being too large for the mentor to evaluate.

This brings up two issues—mentor fatigue and convergence. Since the mentor must listen to all the measures and phrases in a real-time harmonic context,



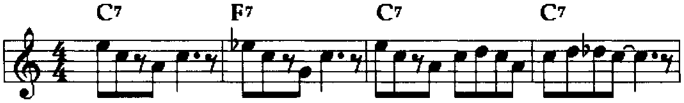
5.2 FIGURE Example phrase.

there is a severe fitness bottleneck (Biles 1994). Unlike interactive genetic algorithms (IGAs) that evolve images, GenJam's individuals cannot be presented in parallel or in a compressed format. The mentor must listen to each individual one at a time, in real time, with suitable accompaniment, which is a very difficult task for most people (Biles 1999). If the populations are too large, the mentor must do a lot of listening for each generation in order to provide fitness values for all the individuals, which means that the evolutionary process will be slower. If populations are too small, however, the tendency for populations to converge on minor variations of a few superindividuals increases, which can lead to highly repetitive solos. For this reason, genetic diversity is critical to GenJam. We will explore this issue further when discussing GenJam's mutation operators.

Figure 5.2 provides a contrived example phrase adapted from Sonny Rollins's "Tenor Madness" to illustrate how GenJam's chromosome structures map to actual notes. Phrase 23 has a fitness of −12, which means it has received 12 more "bads" than "goods" from the mentor. The four measures for this phrase are measure 57, measure 57 again, measure 11, and measure 38. The three referenced measures are shown as individuals in the measure population. Measure 57 has a fitness of 22, which is relatively "good." Its eight events include two rests (the 0s), two holds (the 15s) and four new-note events.

Figure 5.3 shows our example phrase as it would be performed against the first four measures of the chord progression from Figure 5.1. Notice that two of the actual notes in the first two measures differ, even though the same chromosome generated both measures. This is because the chords for these measures were different, which led to different scale mappings, which in turn resulted in different pitches. The dotted quarter note occurs due to the two hold events following the second "7" in the measure individual.

In measure 38, notice that the "7" at the third position in the chromosome maps to D♭ in the last measure of Figure 5.3, which is a note that is not in the scale suggested by a C7 chord. This is the result of a heuristic that maps a



5.3 FIGURE Phrase from Figure 5.2 played against first four bars of tune from Figure 5.1.

repeated new-note event to a chromatic passing tone, as long as the altered note is followed by a new-note event. The intent is to break up sequences of repeated notes and to add some nonscale tones safely. We will see this heuristic again in the description of how GenJam trades fours.

5.4 GENETIC OPERATORS AND TRAINING

The process of creating a new GenJam soloist follows a fairly typical sequence of initialization, selection, crossover, mutation, and replacement. The design of these genetic operators is fairly atypical, however, reflecting the necessity to cope with the fitness bottleneck and convergence issues.

A new soloist is created by initializing the measure and phrase populations with randomly generated individuals. Measure individuals are initialized at the event level, with hold and rest events each occurring with roughly a 20% probability, which is typical for human jazz solos. New-note events are initialized using a fractal generator patterned after Gardner's dice algorithm (Gardner 1978). The process uses a pair of dice, one with 1–7 spots per side, the other with 0–7 spots per side. Summing the spots on the two dice yields numbers in the range 1–14, which is the range needed for new-note events. After rolling both dice for the first new-note event, only one die is rolled for each succeeding new-note, alternating which die is rolled from one to the next. This results in a ramp distribution that peaks in the middle range of the instrument, with horizontal intervals averaging a third and no greater than an octave. This statistically matches the distribution of a mature, trained soloist, which makes the initial generation sound much more musical than if a simple uniform generator is used.

The phrase population is initialized in stages. The first 16 phrases are initialized such that they preserve the fractal sequences in the 64 measures. For example, the first phrase consists of pointers to the first four measures from the measure population. The second phrase consists of the next four measures, and so on. This preserves the fractal sequence across measure boundaries within a phrase and insures that every measure appears in at least one phrase. The remaining 32 phrases are initialized with a uniform generator. Fitness values for both populations are initialized to zero, which means that the mentor should listen and provide feedback for a tune or two before breeding new individuals so that the fitness values have a chance to diverge.

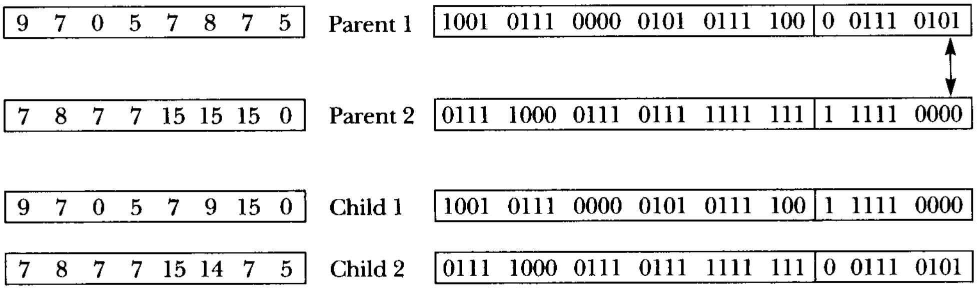
Breeding new generations in both populations begins with a tournament selection/replacement procedure, which chooses four individuals from the population at random to form a "family." The two fittest members of the family are selected to survive and become parents, and their two children replace the two weakest family members back in the population. This proceeds until 50% of each population is replaced. The fitness values for the children are initialized to zero, and they are not allowed to participate in subsequent families in the same generation because they have not been "heard" yet by the mentor. After evaluating a new generation, the mentor decides whether the soloist is good enough to play in public. If not, another generation can be bred.

GenJam also has a "tweak" mode where only the four worst measures and the three worst phrases are replaced by new children. This was developed to address the situation where a soloist sounds good generally but contains a few annoying licks. Tweak mode allows those licks to be replaced without having to replace half of each population. Training a mature soloist typically requires about a dozen generations, followed by a handful of tweak rounds. This usually can be done in a couple of hours.

5.4.1 Crossover

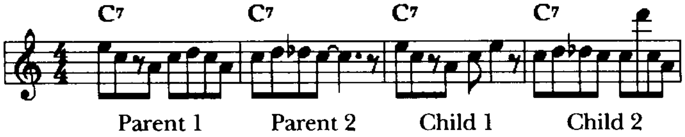
Two new children are bred from their parents by performing a single-point crossover at a random locus in the chromosome pair. The distribution for selecting the crossover point is biased toward the center of the chromosome to promote diversity in the population. If a crossover point is too near one end of the chromosome or the other, the resulting children are more likely to resemble their parents. In the measure population this can lead to small variations of a highly fit measure being overrepresented in the phrase population. In the phrase population, it can lead to the perception that GenJam repeated itself when two nearly identical phrases happen to be played close to one another in the same solo. While human improvisers certainly fall into the trap of overusing their favorite ideas, it didn't seem desirable for GenJam to model *that* aspect of human performance.

The crossover operator behaves identically in both the measure and phrase populations. Figure 5.4 illustrates the crossover operator applied to measures,



Decimal		Binary	
9 7 0 5 7 8 7 5	Parent 1	1001 0111 0000 0101 0111 100	0 0111 0101
7 8 7 7 15 15 15 0	Parent 2	0111 1000 0111 0111 1111 111	1 1111 0000
9 7 0 5 7 9 15 0	Child 1	1001 0111 0000 0101 0111 100	1 1111 0000
7 8 7 7 15 14 7 5	Child 2	0111 1000 0111 0111 1111 111	0 0111 0101

5.4 FIGURE Crossover of measures 11 and 38 from Figure 5.2.



5.5 FIGURE Parents and children from Figure 5.4 played over C7 chord.

using measures 11 and 38 from Figure 5.2 as parents 1 and 2, respectively. Figure 5.5 shows the parents and children played over a C7 chord. In this case, the random crossover point happens to fall within the sixth event of the chromosome, not on an event boundary. This not only mixes the events from the two parent measures, but it also changes the event containing the crossover point in both children. In this case a new-note event in Parent 1 crosses with a hold event from Parent 2. Child 1 inherits a new-note event that is nearly identical to the event at the same locus in Parent 1 (a 9 instead of an 8). Child 2, on the other hand inherits a new-note event that maps to a note over an octave away from the surrounding notes in the measure. Such big intervals seldom sound attractive to most mentors, so Child 2's future looks pretty bleak.

This brings up the reason that 0 was selected to encode a rest, while 15 was selected to encode a hold. Since these events are disproportionately more common than a given new-note event, they are more likely to occur in crossover events. If, for example, 0 and 1 were selected for rest and hold, respectively, their crossover with new-note events would tend to generate low notes, which in turn would tend to drag GenJam's solos into the lower range of its instrument. By encoding rests and holds at the extremes of the coding range, the new-note events created during crossovers with rests and holds are distributed more uniformly. Furthermore, a crossover within a rest and a hold will generate a pair of new new-note events in both children.

5.4.2 Musically Meaningful Mutation

After crossover, one of the two children is selected at random for mutation using one of GenJam's musically meaningful mutation operators. In a simple genetic algorithm with a binary representation, mutation is implemented by flipping an occasional bit, which serves (or not—according to the application at hand) to provide enough diversity over typically hundreds or thousands of generations to insure that the search space is sampled adequately. With GenJam, however, the mentor cannot listen to hundreds or thousands of generations of a soloist—a dozen or two generations is about the limit. This puts enormous pressure on the mutation operators to come up with new individuals that are not just different but clearly better than their predecessors. Consequently, GenJam's mutation operators do considerably more than flip the occasional bit. In fact, GenJam's mutation operators represent most of GenJam's musical intelligence, hence our term "musically meaningful mutation."

Measure Mutation

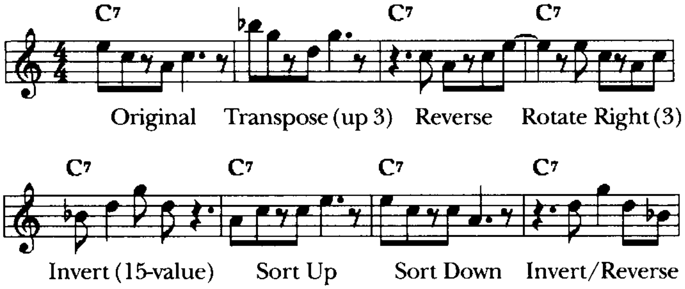
The mutation operators for measures implement adaptations of several standard melodic development techniques—transposition, retrograde, rotation, inversion, sorting, and retrograde-inversion. Because these operators are all musically meaningful, they operate at the event level rather than at the bit level. In other words, these mutations manipulate events, rather than individual bits. The choice of which mutation operator to apply is random.

Table 5.2 illustrates these operators by showing the results of applying them to the chromosome from measure 57 of Figure 5.2. Figure 5.6 shows these chromosomes played against a C7 chord to illustrate their musical effects.

The Transpose operator adds a fixed random increment or decrement to all the new-note events in a measure while ignoring the rests and holds, which preserves the rhythmic structure of the measure while shifting the pitches up or down. In this case the transposition happened to be up three. When applying the transpose operator, GenJam first looks to see how high the measure is and tends to move it in the direction that has the most room to shift without hitting a

Original Measure	9	7	0	5	7	15	15	0
Transpose (up 3)	12	10	0	8	10	15	15	0
Reverse	0	15	15	7	5	0	7	9
Rotate Right (3)	15	15	0	9	7	0	5	7
Invert	6	8	15	10	8	0	0	15
Sort Up	5	7	0	7	9	15	15	0
Sort Down	9	7	0	7	5	15	15	0
Invert/Reverse	15	0	0	8	10	15	8	6

5.2 TABLE Mutations of measure 57 of Figure 5.2.



5.6 FIGURE Mutated measures from Table 5.2 played against a C7 chord.

ceiling or a floor. If an event is shifted outside of the new-note range of 1–14, it is reflected back into that range.

The Reverse operator reverses the events in the measure. This is similar to retrograde, a compositional technique where the notes are played in reverse order. The difference is subtle, but can be seen in the first note of the reversed measure, which was shortened from a dotted quarter note to an eighth note followed by rests. Inspection of the chromosomes reveals that the holds that created the dotted quarter note in the original measure are holding a rest in the reversed measure.

The Rotate Right operator rotates the events to the right a random amount (three positions in the example). In this case the rotated measure begins with two hold events, which has the effect of holding the last note of the previous measure, resulting in the tied E that begins the rotated measure. Such ties can be interesting when a note is held across a measure boundary and the chord in the new measure suggests a scale that does not include the held note. This results in a dissonance that tends to resolve when the first new-note event of the new measure is played.

The Invert operator subtracts each event from 15. The effect is that rests become holds, holds become rests, and the melodic contour of the measure is turned upside down. This is similar to inversion, another standard compositional technique, but like the reverse operator, rhythm is also affected, as can be seen by inspecting the resulting measure in Figure 5.6.

The two Sort operators ignore rest and hold events and sort the new-note events in the measure in either ascending or descending order. The effect is to retain the rhythmic structure of the measure but to replace the melodic contour with an ascending or descending melodic line. This operator can smooth out a measure with lots of jumps, but it often creates sequences of repeated notes, which can sound uninteresting. After sorting, the operator looks for sequences of two or more identical new-note events and attempts to create passing tones. This heuristic will appear again in the discussion of real-time interaction.

Finally, the Invert/Reverse operator performs both the Invert and Reverse operators. This is similar to retrograde-inversion, yet another standard compositional technique. Once again, rhythm is affected, as can be seen by inspecting the resulting measure. This operator is very useful in generating fresh material and will appear again in an expanded form in the section on real-time interaction.

Phrase Mutation

The mutation operators on phrases are similar to those on measures in that they are musically meaningful and operate on the pointer level rather than the bit level. Table 5.3 illustrates the phrase operators by showing the results of applying them to the chromosome from phrase 23 of Figure 5.2. The Reverse and Rotate Right operators behave the same with phrase individuals as they did with measure individuals.

One of the techniques many improvisers use to develop melodic ideas is "sequencing," where the improviser repeats a melodic motive with variations on the repetitions. The Sequence Phrase operator implements this technique by picking a random measure pointer in the phrase chromosome and replacing it with one of its neighbors. As we have already seen with the repetition of measure 57 in Figure 5.2, the same measure played over different chords tends to result in different pitches, which makes the repetition sound like a development of the idea rather than merely a repetition of it.

Original Phrase	**57**	**57**	**11**	**38**
Reverse	38	11	57	57
Rotate Right (3)	57	11	38	57
Sequence Phrase	57	57	*38*	38
Genetic Repair	57	57	11	*29*
Super Phrase	41	16	57	62
Lick Thinner	*31*	57	11	38
Orphan Phrase	17	59	43	22

5.3 TABLE Mutations of phrase 23 of Figure 5.2.

The Genetic Repair operator replaces the least fit measure in the phrase with a randomly selected one. The intent is to "repair" a phrase that has one "bad" measure. Similar but more extreme, the Super Phrase operator discards the child created from crossover and creates a brand-new phrase in its place consisting of the winners of four three-measure fitness tournaments. The intent is to create phrases that consist of "winner" measures in the hope that combining better measures will produce a better phrase. In the example, measure 57 happened to win the third tournament.

The Lick Thinner operator attacks the convergence problem by replacing the measure in the phrase that occurs most frequently in the entire phrase population with a measure that occurs infrequently. The intent is to reduce the oversampling of highly fit measures by replacing such measures with ones that are undersampled. The Orphan Phrase operator takes this idea to an extreme by creating a brand-new phrase of the four measures that occur least frequently in the entire phrase population. The intent is to insure that every measure appears in at least one phrase. An orphan measure, one that does not appear in any phrase, cannot be heard by the mentor. If its fitness is sufficiently high, it could never be replaced, and the chances of it reappearing in a phrase are slim, which is an inefficient use of an individual in the measure population. By creating a family for the orphan, it can once again be heard by the mentor and either survive or be replaced accordingly.

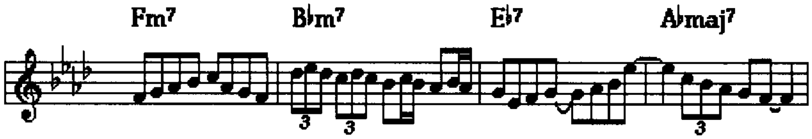
One final note on replacement in both populations is in order. After the two children are created, one by crossover alone and the other by both crossover and mutation, they are compared to every member of the population. If a child resembles any existing member of the population too closely (at least 75% of the events or pointers are identical), it is mutated until it is sufficiently unique before being placed in the population. This is yet another attempt to chip away at the tendency of the evolutionary algorithm machinery to converge by increasing diversity in the populations.

5.5 REAL-TIME INTERACTION

The most engaging aspect of GenJam's performance is its ability to trades fours and eights in chase choruses. GenJam listens to the human performer's four using a pitch-to-MIDI converter and maps what it thought it heard to four measure chromosomes that are separate from the measure population. A phrase chromosome is also created that simply references those four measures in the order in which they were heard. In the last instant of the human's four, GenJam stops listening and often applies its mutation operators to the phrase and four measure chromosomes it has just built. The mutated chromosomes are then played back against the chords of the next four measures of the tune as GenJam's response. In other words, GenJam evolves what it just heard into what it then plays. After a brief explanation of the hardware GenJam uses, we will explore in detail a sequence of fours played by the author and GenJam.

GenJam uses a Roland GI-10 guitar-MIDI interface as a pitch-to-MIDI converter. Although the GI-10 was designed as a guitar interface, the Roland engineers were thoughtful enough to include a microphone input on the off chance that someone might use it for vocals or acoustic instruments like the author's trumpet and flugelhorn. The author uses a cheap microphone with a low input level and poor frequency response, which he positions close to the bell of the trumpet. The poor frequency response acts as a band-pass filter, and the low input level tends to filter out ambient noise so that only the trumpet is heard by the GI-10. As we shall see, the GI-10 makes a lot of mistakes, which would make it unsuitable for most pitch-tracking applications, but GenJam's robust representation scheme serves as the perfect target for the GI-10, and the result is a highly fault-tolerant system.

The example fours come from measures 25–32 of Jerome Kern's "All the Things You Are," a standard much loved by jazz musicians, and they were generated at a "rehearsal" of the Virtual Quintet. In measures 25–28, the author decided to play a quote from "Prince Albert," a Charlie Parker line composed over the chords to "All the Things You Are." GenJam's response in measures 29–32 was particularly nice, although not atypical. Figure 5.7 shows the "Prince Albert" quote played by the author on trumpet. You'll have to trust the author's word that he played the quote faithfully. A cursory glance at Figure 5.7 reveals that the quote is not restricted to eighth-note multiples, meaning that GenJam cannot



5.7 FIGURE "Prince Albert" lick played over measures 25–28—what the human played.

possibly represent this phrase accurately, even if it received the correct MIDI events from the GI-10, which it didn't.

Table 5.4 represents the MIDI events sent by the GI-10 to GenJam, as recorded in a data dump made during the performance of the tune. Each row represents an eighth-note length window, and the left column denotes the measure number and window within the measure, using offsets instead of indices as labels. The columns represent distinct MIDI events received during each window. Negative numbers represent MIDI note-off events, where the number is the MIDI pitch number for the note being turned off. The number 15 in parentheses indicates that no MIDI activity was received in that window, which maps to a hold event (15) in the chromosome. This occurs in windows (2, 4), (3, 0) and (3, 6).

Triples are note-on events, with the first number indicating the MIDI pitch, the second number the MIDI velocity (loudness), and the third number GenJam's new-note event number. This is derived by looking up the received MIDI pitch in the scale suggested by the chord being played by the rhythm section and selecting the offset in that scale whose corresponding pitch is closest to the received pitch without exceeding it. In other words, GenJam runs its play routine in reverse by looking up the nearest pitch in the scale it would use if it were playing instead of listening, and it uses the offset of that pitch as the new-note event number.

It is obvious that the GI-10 hears more notes than the human played. These excess events are handled by simply remembering the last note-on event that occurred in a window and ignoring those that might have come before it. For a given window, the GenJam new-note event number that made it into a measure chromosome is the last number that appears in that row and is indicated with bold italic in Table 5.4. If a window receives only a MIDI note-off event, it maps to a GenJam rest event, as is the case in windows (3, 1) and (3, 7).

The chromosomes built from this stream of MIDI traffic appear in Figure 5.8, with the phrase chromosome appearing vertically as the leftmost column and the four measures appearing horizontally next to their references. We refer

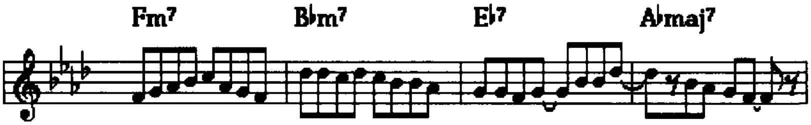
M W	Event	Event	Event	Event	Event	Event	Event	Event
0 0	−68	67 127 10	−67	65 117 ***9***				
0 1	−65	68 100 11	−68	67 113 ***10***				
0 2	−67	68 127 ***11***						
0 3	−68	70 101 ***12***						
0 4	−70	72 127 13	−72	71 127 13	−71	72 127 ***13***		
0 5	−72	68 107 ***11***						
0 6	−68	67 117 ***10***						
0 7	−67	65 114 ***9***						
1 0	−65	73 127 ***14***						
1 1	−73	74 127 14	−74	75 127 14	−75	73 127 ***14***		
1 2	−73	73 127 14	−73	72 127 ***13***				
1 3	−72	72 99 13	−72	73 113 14	−73	73 120 ***14***		
1 4	−73	72 123 13	−72	71 127 ***13***				
1 5	−71	70 118 ***12***						
1 6	−70	70 116 12	−70	72 121 13	−72	70 115 ***12***		
1 7	−70	68 121 ***11***						
2 0	−68	68 88 12	−68	70 107 12	−70	68 107 12	−68	67 101 ***11***
2 1	−67	67 31 GN	−67	67 65 ***11***				
2 2	−67	63 96 9	−63	63 35 GN	−63	63 115 9	−63	65 111 ***10***
2 3	−65	66 89 11	−66	67 106 ***11***				
2 4	(***15***)							
2 5	−67	68 101 ***12***						
2 6	−68	69 107 12	−69	70 114 ***12***				
2 7	−70	74 90 14	−74	75 108 ***14***				
3 0	(***15***)							
3 1	−75 0 ***0***							
3 2	72 127 13	−72	71 127 13	−71	71 95 13	−71	70 104 ***12***	
3 3	−70	68 105 ***11***						
3 4	−68	67 106 ***10***						
3 5	−67	67 98 10	−67	65 99 ***9***				
3 6	(***15***)							
3 7	−65 0 ***0***							

5.4 TABLE MIDI events sent by pitch-to-MIDI converter—what GenJam received.

0	9	10	11	12	13	11	10	9
1	14	14	13	14	13	12	12	11
2	11	11	10	11	15	12	12	14
3	15	0	12	11	10	9	15	0

5.8 FIGURE

Phrase and four measure chromosomes before mutation—GJNF of human's four.



5.9 FIGURE

Phrase in Figure 5.8 played over original chords—what GenJam heard.

to this as the GenJam Normal Form (GJNF) of the human's four. Figure 5.9 shows the phrase from Figure 5.8 played against the chords of measures 25–28. This is essentially what GenJam "heard" from the human. It retains the overall contour of the human's four, but it is far from an accurate transcription. This is not a problem, however, because the ultimate goal is to develop what the human played, not play it back note for note. An optimistic but accurate characterization is that a mistake in pitch tracking is a form of melodic development, not an error!

If the phrase from Figure 5.8 were played over the chords of measures 29–32, which is what GenJam will do with its response, the result would be a rather uninteresting parroting of the human's four. Mining another soloist's four is only part of the battle—one needs to develop what was mined into a creative response. GenJam accomplishes this by applying its mutation operators to the phrase and four measure chromosomes. Not all the mutation operators described in Section 5.4 are used in trading fours, and some new ones are used in trading fours that are not used in evolving soloists.

The mutations from Table 5.2 that are used on measures include the Transpose, Reverse, Rotate, Invert, and Invert/Reverse. The Sort operators were dropped because they tended to produce large jumps at measure boundaries. The Invert operator was modified for the same reason by transposing the inverted measure so that it matches the range of pitches in the original measure. This addresses the problem where the human plays a four in the upper register, for example. Inverting a measure in that four will put GenJam's response for that measure in the lower register, which will sound unfortunate if the other measures in the phrase remain in the upper register. By matching the range of the original measure, the inversion has a better chance of fitting well with its neighboring measures.

This brings up a general problem in stretching an IGA to a real-time setting—there is no time for a mentor to provide fitness values. For GenJam to be responsive, it must apply its mutation operators in about 30 milliseconds and play the result immediately. If the human plays a nice four, GenJam should respond immediately with a nice four, and there are no second chances. This means that the mutations must be more than musically meaningful—they also must be safe because whatever they produce will go out unevaluated. Eliminating fitness from this process calls into question whether it is still evolutionary in nature, an issue we shall address in the conclusion to this chapter.

Of the phrase operators used to train soloists, only the Reverse, Rotate, and Sequence operators are appropriate here. The others address optimization and convergence issues, which are irrelevant when there is only one phrase and four measures in the "population."

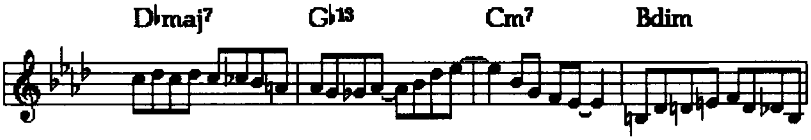
GenJam also uses three special operators in trading fours that combine measure and phrase mutations. The first of these is a phrase-level reverse that reverses the phrase *and* all four of the measures, resulting in a four that is essentially the human's four played backward. This could not be done when training a soloist because reversing the measures for a given phrase would disrupt other phrases that used those measures, but here there is no problem because the mutated measures are in only one phrase and will be discarded after the four is played. The second special mutation is the same as the first, except all four measures are also inverted, using the range-corrected inversion operator described above. This results more or less in a retrograde inversion of the entire phrase.

The third special mutation is an extended sequence operator that uses the last measure of the human's four as the first measure of the response, copies that measure over the second and possibly the third measure of the human's four, uses one or two of the original first three human measures for the remaining measure(s) in the phrase, and mutates at least the repeated measures. This produces a phrase that quotes the human's last idea and then develops it two or three times before finishing with other material.

The actual mutations chosen in our example were minimal. Measure 0 was transposed down two steps, and the phrase was rotated right three positions, which has the same effect as rotating left one position. A heuristic was also applied where the repeated 12 in measure 2 was changed to 13 because it then formed a passing tone. This is an extension of the chromatic passing tone heuristic described above and is used to break up repeated notes in safe ways. Figure

0	14	14	13	14	13	12	12	11
1	11	11	10	11	15	12	***13***	14
2	15	0	12	11	10	9	15	0
3	***7***	***8***	***9***	***10***	***11***	***9***	***8***	***7***

5.10 FIGURE Phrase and four measure chromosomes after mutation.



5.11 FIGURE Phrase in Figure 5.10 played against measures 29–32—what GenJam played.

5.10 shows the chromosomes after mutation, with the altered values in bold italic.

Figure 5.11 shows the phrase in Figure 5.10 played over the chords of measures 29–32. This was GenJam's response to the human's "Prince Albert" quote. Notice the chromatic passing tone heuristic in action once again, particularly in the nice eighth-note chromatic run from the second D♭ in the first measure of the four through the G♭ in the second measure. The mutations were minor at the chromosome level but the resulting four was very different and quite nice.

In performance GenJam is a formidable opponent when trading fours. It is able to catch and develop phrases better than any human the author has run across in the thousand or so jam sessions he has played over the years. Musical conversations with GenJam are great fun for the performer, and that translates into a more energetic and engaging performance. Before adding real-time interactivity to GenJam about two years ago, gigs with GenJam could be exhausting because the creativity was primarily on the human's shoulders. Gigs are now a breeze because GenJam is a stimulating collaborator that exhibits true creativity.

5.6 CONCLUSIONS

The Virtual Quintet plays in public two or three times per month, on the average. The types of performance run the gamut from background to foreground situations and include providing background music at receptions and parties, giving lecture/demonstrations, playing at coffeehouses, and giving an occasional concert. The author has not had the guts to go into a real jazz club yet, although the quintet's repertoire is more than sufficient to handle such a gig. The major flaw with GenJam in a foreground setting like a jazz club is the lack of visual impact. In a foreground setting the audience expects to see as well as hear the interaction among performers, and while there certainly is interaction between the author and GenJam, that interaction is entirely aural, not visual. Furthermore, when GenJam takes a full-chorus solo, what should the human performer do? Tapping one's foot, bobbing one's head, and otherwise grooving on what GenJam is playing is a bit contrived if not downright silly. In background settings, this is not a problem because the audience does not expect to be entertained in the same way the audience in a jazz club does. During GenJam's solos in background settings, the author tends to stroll around the room, get a drink, or chat with people who seem interested. In a jazz club, such behavior is clearly not appropriate, Miles Davis not withstanding . . .

In an attempt to engage the audience during GenJam solos in concerts, the author has tried several experiments in audience-mediated performance (Biles and Eign 1995). Audience members are given feedback paddles, which are red on one side and green on the other. The audience is instructed to show the green side to the stage when they like what GenJam is currently playing, show the red side when they don't like what GenJam is playing, and show nothing if they feel neutral or don't care. The author then plays several training tunes, on which GenJam solos for three or more choruses, and types "g"s (good) and "b"s (bad) from the stage based on how much green or red he sees. After four to five tunes, the audience usually starts to get restless, and the author then rejoins the quintet to play some tunes with the audience's soloist. In this way the audience influences the content of the performance and hopefully better understands what GenJam is doing. The typical scenario for performing a set in a concert setting is to play a few tunes using pretrained soloists to start the concert, do the training experiment, and then play some tunes with the audience's soloist. This works well for about an hour in a concert setting, and the author has performed such concerts in several cities. One charming feature of these concerts is the universal phenomenon of the audience showing feedback cards during the author's first solo after the training tunes. Thankfully, the author tends to see more green than red in these situations!

We conclude with a brief discussion on the role of fitness in GenJam. Fitness is generally considered a key component of an evolutionary system, but as we have seen, GenJam eliminates the need for fitness when trading fours by employing mutation operators that "always work." One perspective on this is to view evolutionary computation as a generate-and-test strategy. To put it simply, if the generators are smart enough, testing is redundant. This is the case when GenJam trades fours, but only if the human performer gives GenJam good fours to mutate. In such a case, however, one could argue that the search is so directed as to be not a search at all, but rather a direct, albeit stochastic mapping from problem to solution. This is the logical endpoint for GenJam's evolution because what started as a search of a huge and ill-understood melodic space ends up as a process that reliably develops human melodic fragments in pleasing ways. Another way to view this is that, at least when trading fours, GenJam has evolved from an evolutionary system to a knowledge-based system, with its mutation operators representing its knowledge.

The critical component here is the input to the process. When evolving a soloist, GenJam starts with random initial populations, which, fractal initialization notwithstanding, are not likely to sound particularly musical. When trading fours, GenJam's "populations" are the phrase and four measures from the human performer, which is not random and hopefully is musical. In a sense, then, GenJam depends as much on the performer's ability to play good fours as it depends on the mentor's ability to provide good fitness. GenJam, then, is still an interactive evolutionary algorithm when it trades fours, but the interaction occurs in providing the populations, not the fitness. In the end, GenJam's creativity still depends, at least in part, on the stimulation provided by its fellow performers, just like any other jazz musician.

REFERENCES

Biles, J. A. (1994). GenJam: A Genetic Algorithm for Generating Jazz Solos. In *Proceedings of the 1994 International Computer Music Conference,* ICMA, San Francisco.

Biles, J. A. (1998). Interactive GenJam: Integrating Real-Time Performance with a Genetic Algorithm. In *Proceedings of the 1998 International Computer Music Conference,* ICMA, San Francisco.

Biles, J. A. (1999). Life with GenJam: Interacting with a Musical IGA. In *Proceedings of the 1999 IEEE International Conference on Systems, Man, and Cybernetics,* Tokyo.

Biles, J. A., P. G. Anderson, and L. W. Loggi (1996). Neural Network Fitness Functions for a Musical IGA. In *Proceedings of the International ICSC Symposium on Intelligent Industrial Automation (IIA'96) and Soft Computing (SOCO'96)*, March 26–28, Reading, UK, ICSC Academic Press, pp. B39–B44.

Biles, J. A. and W. Eign (1995). GenJam Populi: Training an IGA via Audience-Mediated Performance. In *Proceedings of the 1995 International Computer Music Conference,* ICMA, San Francisco.

Coker, J. (1964). *Improvising Jazz.* Prentice-Hall.

Dannenberg, R. B. (1993). *The CMU MIDI Toolkit, Version 3.* Carnegie Mellon University. *http://www.cs.cmu.edu/~rbd/.*

Gannon, P. (1991). *Band-in-a-Box.* PG Music, Inc., Hamilton, Ontario, 1991–1999. *http://pgmusic.com/.*

Gardner, M. (1978). White and Brown Music, Fractal Curves and One-over-f Fluctuations. *Scientific American* 238(4):16–27.

Haerle, D. (1980). *The Jazz Language.* Studio 224.

Haerle, D. (1989). *The Jazz Sound.* Hal Leonard.

Levine, M. (1995). *The Jazz Theory Book.* Sher Music Company.

Russell, G. (1959). *The Lydian Chromatic Concept of Tonal Organization for Improvisation.* Concept Publishing.

Sabatella, M. (1992). *A Jazz Improvisation Primer.* Outside Shore Music. *http://www.outsideshore.com/primer/primer/.*

This Page Intentionally Left Blank

6 CHAPTER On the Origins and Evolution of Music in Virtual Worlds

Eduardo Reck Miranda Sony Computer Science Laboratory, Paris

6.1 INTRODUCTION

We are investigating how musical forms may originate and evolve in artificially created worlds inhabited by virtual communities of musicians and listeners. Origins and evolution are not, however, studied here in the context of genetic mutations, but rather in the context of cultural conventions. We consider that variations in musical styles, for example, result from the emergence of new rules and/or from the shifting of existing conventions for music making (the term "music making" is used here to mean both creating music and listening to music). Musical styles maintain their organization within a cultural framework and yet they are highly dynamic; they are constantly evolving and adapting to new cultural situations (Reck 1997). While the criteria for natural selection in biology are chiefly based upon physical fitness and reproductive capability, in music these criteria will depend upon on the effects of the new rules on the music-making experience.

In this context, musical forms can be studied as live organisms, and as such musicology can benefit hugely from artificial life's (A-Life) research paradigms. Particularly interesting paradigms that have emerged recently include *evolutionary algorithms* in their various forms (Schwefel 1965; Fogel, Owens, and Walsh 1966; Holland 1975; Goldberg 1989; Koza 1992), *cellular automata* (Ermentrout and Edelstein-Keshet 1993; Wolfram 1994), and *autonomous robots* (Steels and Brooks, 1995), to cite but a few. Indeed, several musicological investigations have already begun to look closely at these developments, notably by composers interested in the organization principles that underlie A-Life's pattern generation algorithms (Degazio 1997; McAlpine, Miranda, and Hoggar 1999; Miranda 1993, 1994). We have come to believe, however, that these paradigms should be considered as if they were fulfilling particular roles within a much more complex phenomenon, namely, *social evolution.*

The first half of this chapter introduces the theoretical background of our research. Here we discuss the general notion of evolutionary modeling and propose the fundamental ingredients for studying evolutionary phenomena in the context of music. Then we focus on the design of a digital musical instrument, where we have employed cellular automata to control the evolution of complex synthesized sounds. Finally, we discuss the relevance of this musical instrument in the context of the overall research project.

6.2 EVOLUTIONARY MODELING

Music can be modeled as an adaptive system of sounds used by a number of individuals (or *distributed agents* in computer science jargon) engaged in a collective music-making experience; some may only listen to the sounds ("audience"), while others may be fully engaged in the generative process ("musicians").

Our hypothesis is that music emerges from the interaction of the agents when they engage in such music-making experiences. In this case, there is no global supervision taking place, and the agents do not have direct access to the musical knowledge of the other agents, apart from hearing what they actually do during the interactions. This model could be metaphorically compared to a drumming session, where people, who have not necessarily met before, can join in and play, or simply watch. A contrasting scenario would be an orchestral concert where musicians follow a common score under the direction of a conductor; there is very little room for music evolution in this scenario.

On the opening pages of the first issue of *Evolution of Communication Journal,* Luc Steels proposes four fundamental mechanisms for studying the origins of language: *evolution, coevolution, self-organization,* and *level formation* (Steels 1997). For the sake of clarity, in this essay we have replaced the term "evolution," as originally used by Steels, by the expression "transformation and selection." As these mechanisms also seem to work well for the study of other cultural phenomena, we have taken them as the starting point of our research into the origins of music.

Before we discuss these mechanisms, it is important to clarify the contexts in which the term "evolution" appears in our research. In natural history, evolution is frequently associated with the idea of transitions between species. The Darwinian argument that humans originated from anthropoid apes is a typical example of this approach to evolution. Indeed, this approach made a serious impact on

19th-century cultural anthropology, which maintained that humankind has become increasingly sophisticated during the linear course of history. The legacy of this argument is the general popular belief that the Stone Age, for example, represents far less sophistication than the Iron Age. This approach to evolution suffers from taking for granted that one can "measure" evolution entirely based on materialistic criteria. For example, one should not take for granted that European classical music is more sophisticated than African drumming solely based on the technological sophistication of the instrument used. The rudimentary technological development of most non-European societies (e.g., Pygmies) gave rise to very sophisticated social systems and complex religious rituals. A remote community living in an environment where prey and crops abound would not be under pressure to develop sophisticated hunting weaponry or irrigation technology, for instance. These people would probably give priority to the creation of a belief system in which a religious ritual would have to be performed occasionally in order to maintain the richness of their habitat.

In the context of our research, the notion of evolution denotes the concept of transition from one state of affairs to another. This transition is not therefore necessarily associated with the idea of improvement, but rather with the notion of *increase in complexity*.

The fundamental mechanisms for studying the origins and evolution of music are discussed below.

6.2.1 Transformation and Selection

When a transformation process creates variants of some type of entity, normally there is a mechanism that favors the best transformations and discards those that are considered inferior, according to certain criteria. For example, the criterion in biology might be fitness for survival, while in linguistics the selection of an utterance could involve a compromise between the effortless use of the speaker's articulatory mechanism and the degree of understanding by the listener. What is important here is that these transformations must *preserve the information* of the entity; otherwise the entity is destroyed.

In music, the selection of sound objects is based upon psychoacoustics (e.g., the interplay between repetition and variation) and physical criteria (e.g., the capability of the musical instrument(s) available). These selection criteria are the crux of the design of realistic evolutionary models for music.

A plausible methodology for studying evolutionary mechanisms in linguistics has been proposed by Steels and coworkers (de Boer 1997; Steels and Vogt 1997). Inspired by Wittgenstein's concept of language games (Wittgenstein

1963), they have devised a number of different games for investigating different processes of language formation: phonology, meaning, syntax, lexicon, and so forth. We have been employing similar games to study the origins of musical systems, such as pitch systems, for example (Miranda 1999).

6.2.2 Coevolution

Coevolution involves the interaction of various contiguous transformation and selection processes. Selection criteria are not fixed, but rather they are affected by an environment that is also in a state of flux. While evolution tends to drive a system toward the improvement of particular aspects, coevolution tends to push the whole system toward greater complexity in a *coordinated manner.* For example, the evolution of musical styles should normally be studied in close association with the evolution of musical instruments, and vice versa.

As an example, consider the case of the keyboard class of instruments: the evolution of the piano is popularly associated with the settlement of the equal-temperament tuning system (a tuning system in which all notes of the scale are equally separated by exactly half a tone) and with the increasing use of expressive dynamics in compositions. The piano can produce a much wider band of variations in dynamics than the harpsichord, from extremely loud to extremely quiet notes. On the one hand, the equal-temperament system alleviated the problem of tuning different instruments in an ensemble, but on the other hand, it increased the scope for composers to explore much more complex harmonic structures in their music; for example, the modulation to different tonal keys within the same piece (Campbell and Greated 1987).

6.2.3 Self-organization

The notion of self-organization is closely related to the notion of *coherence.* The origins of coherence in distributed systems with many interacting agents is a vast research topic. To put it in simple terms, three ingredients are needed for self-organization to take place in a system:

- A set of possible variations
- Random fluctuations
- A feedback mechanism

Random fluctuations in the system will eventually strengthen some of the fluctuations because of the feedback mechanism: the more a fluctuation is strengthened, the more predominant it becomes. As an example, imagine a musical game as follows: a group of virtual agents engage in a drumming session, but none of the agents is an experienced musician; that is, they have no musical training. Each agent can bring in a different percussion instrument, but an agent must never have played its instrument before. They all agree that they should start playing sounds simultaneously in any way they wish. To begin with, they will certainly produce a highly disorganized mass of rhythms; in this case we say that the system is in equilibrium. The set of possible variations in this system is the set of all noises and rhythms that can be produced by the instruments. Next, imagine a situation in which at some point an agent A makes a very distinct sound pattern that catches the attention of a fellow agent B. The fellow agent then attempts to imitate it. The imitation may not be exact (for example, because agent B's instrument is different from agent A's), but agent A recognizes it as an imitation of the pattern and instinctively reproduces the original pattern again; that is, agent A gives a positive feedback to agent B. The other agents will probably be keen to imitate this pattern as well, and variations will certainly start to emerge. After a while, the pattern that was originated by agent A, plus its variations, should become successful conventions. The next time these agents engage in a jam session they will certainly remember these and other patterns, and will probably play them during the session. The more the patterns are played, the more conventional they become. When not engaged in jam sessions, some agents may even consider adapting their instrument to better produce those patterns, while others will spend a great deal of effort practicing ways to produce them.

The musical plot described above would equally apply to studying the emergence of spoken language, for example. In this case the agents would be engaged in linguistic communication. The need for explicit communication of rational meanings would exert pressure for the formation of different types of syntactical conventions. The intention of the social interaction defines the rules of the game: for example, in a linguistic situation, it may be better that the agents engage in a one-to-one interaction rather than in a simultaneous one. Also, the nature of the evolutionary phenomena may differ according to the intention of the game. For instance, an utterance that is difficult to produce due to the limitations of the vocal tract will have low priority for becoming part of a language, whereas in music such a limitation would be much relaxed because the agents could, for example, modify their musical instruments or even create new ones.

Most cultural phenomena seem to follow identical self-organizing principles in some way; see for example an interesting account of the origins of the flamenco song style in a paper by Washabaugh (1995) that appeared in *Journal of Musicological Research.*

6.2.4 Level Formation

By "level formation" we mean the formation of *higher-level conventions,* such as both semantics and syntactical conventions in language, for example.

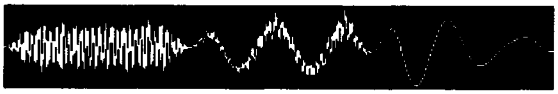
Suppose that at some point in the musical scenario described above, agents start remembering rhythmic patterns in terms of repeated short sequences grouped together as units; for example, shorter sounds may be grouped by similarity in duration and proximity in time, and repeated patterns may be grouped as units based on their parallel structure. This figurative conceptualization of rhythm should then yield more abstract conceptualizations such as metric rules and sense of hierarchical functionality; for example, the concept of strong beat versus weak beat (Bamberger 1991). Multiple rhythmic conventions for groupings and hierarchical organizations would soon start to emerge in the community of agents. Level formation is the least understood of the four mechanisms; much research is needed on this front.

6.3 EVOLVING SOUND WITH CELLULAR AUTOMATA

Computer sound synthesis technology has enabled musicians to control sound at its most fundamental levels, from the fine-tuning of its spectral components to the dynamics of unfolding. Perhaps one of the most interesting sound synthesis techniques available today is the *granular synthesis* technique (Miranda 1998a). Granular synthesis of sounds involves the production of thousands of tiny sound granules (e.g., 35 milliseconds long) that combine to form complex and dynamic sounds (Figure 6.1).

The sounds from granular synthesis tend to exhibit a great sense of movement and flow. This synthesis technique can be metaphorically compared with the functioning of a motion picture in which an impression of continuous movement is produced by displaying a sequence of slightly different images at a rate beyond the scanning capability of the human eye.

The composer Iannis Xenakis is commonly cited as one of the mentors of granular synthesis. In the 1950s, Xenakis (1971) developed important theoretical writings where he laid down the principles of the technique. The first computer-based granular synthesis systems did not appear, however, until Curtis



6.1
FIGURE
A sequence of three different tiny sounds. A rapid succession of thousands of tiny sounds forms larger complex sounds.

Roads (1991) and Barry Truax (1988) began systematically to investigate the potential of the technique.

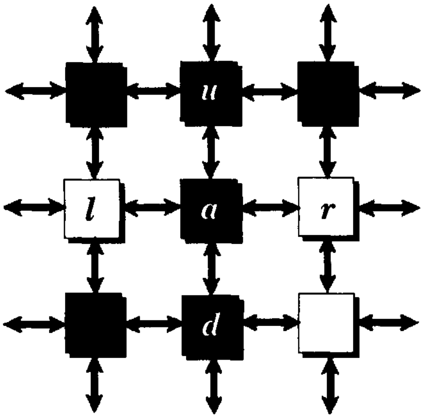
The main difficulty of granular synthesis for musicians is the specification of the nature of each individual sonic granule and how it will affect the overall result. So far, granular synthesis systems have used stochastic formulae (i.e., probabilities) to control the evolution of the granules; for example, to control the waveform and the duration of the individual granules as they evolve in time. We propose that the self-organization principles of cellular automata are more adequate to control the evolution of these granules than stochastic formulae.

6.3.1 The Basics of Cellular Automata

Cellular automata (CA) are computer modeling techniques widely used to model systems in which space and time can be represented discretely and quantities take on a finite set of discrete values.

CA were originally introduced in the 1960s by von Neumann and Ulan as a model of biological self-reproduction (Cood 1968). They wanted to know if it would be possible for an abstract machine to reproduce; that is, to automatically construct a copy of itself. Their model consisted of a two-dimensional grid of cells (i.e., variables), each cell of which had a number of states (i.e., values for the variables), representing the components out of which they built the self-reproducing machine. Controlled completely by a set of rules designed by its creators, the machine would extend an "arm" into a virgin portion of the grid, then slowly scan it back and forth, creating a copy of itself—reproducing the patterns of the cells at another location in the grid.

Since then CA have been repeatedly reintroduced and applied for a considerable variety of modeling purposes (see, for example, Wolfram 1994). Many interesting algorithms have been developed during the past 30 years. In general, CA are implemented on a computer as a regular array or matrix of cells; they normally can have one, two, or three dimensions. Each cell may assume values from a finite set of integers, and each value is normally associated with a color. The functioning of a cellular automaton is displayed on the computer screen as



6.2 FIGURE An example of a matrix of cells where the transition function computes the value for a cell at time $t + 1$ based upon the values of its four neighbors.

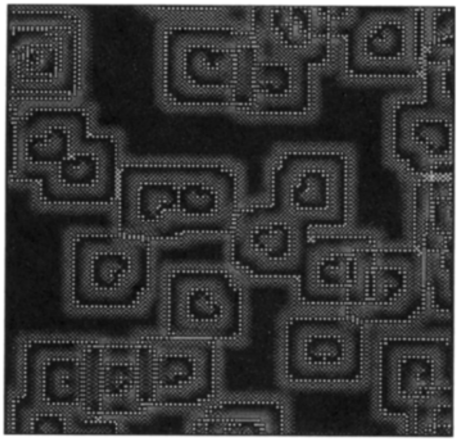
a sequence of changing patterns of tiny colored cells, according to the tick of an imaginary clock, like an animated film. At each tick of the clock, the values of all cells change simultaneously, according to a set of transition rules that takes into account the values of their neighborhood. Figure 6.2 portrays an example of a matrix of cells where the transition function computes the value for a cell at time $t + 1$ based upon the values of its four neighbors: u, d, r, and l; in this case, an example rule could be "A cell will have value 1 (black) at time $t + 1$ if at least 2 neighbors have value 0 (white) at time t."

The set of the values of all cells at a specific lapse of time t is called a configuration c^t. Given an initial configuration c, a global transition function F determines a sequence $c, c^1, c^2, \ldots, c^n$, called a propagation, where $c^{t+1} = F(c^t)$ for all t.

6.3.2 The Cellular Automaton Used in Our System

The cellular automaton used in our system is called ChaOs (an acronym for Chemical Oscillator). ChaOs is an adapted version of a cellular automaton that has been used to model the behavior of a number of oscillatory and reverbatory phenomena such as Belouzow-Zhabotinsky-style chemical reactions (Dewdney 1988).

ChaOs can be thought of as a matrix of cells containing identical electronic circuits. At a given moment, cells can be in any one of the following states: quiescent, depolarized, or collapsed. The automaton tends to evolve from an initial



6.3 FIGURE The ChaOs CA in action.

random distribution of states in the grid of cells toward an oscillatory cycle of patterns (Figure 6.3).

The metaphor behind the ChaOs model is as follows: A cell interacts with its neighbors (normally eight neighbors) through the flow of electric current between them. There are minimum (V_{min}) and maximum (V_{max}) threshold values that characterize the state of a cell. If its internal voltage (V_i) is below V_{min}, then the cell is "quiescent." If it is between V_{min} (inclusive) and V_{max} values, then the cell is being "depolarized." Each cell has a couple of resistors (*R*1 and *R*2) aimed at maintaining V_i below V_{min}. But when it fails (that is, when V_i reaches V_{min}), then the cell becomes depolarized. There is also a capacitor (k) that regulates the rate of depolarization. The tendency, however, is to become increasingly depolarized with time. When V_i reaches V_{max}, the cell collapses. A "collapsed" cell at time t is automatically restored to a quiescent state at time $t + 1$.

Assuming that M is a discrete set of n states m_p, such that $n \geq 3$, then we can define the states of the cells as follows:

- If $V_i < V_{min}$, then m_0 (i.e., quiescent).
- If $V_i \geq V_{min}$ and $V_i < V_{max}$, then m_p (i.e., depolarized), where $p > 0$ and $p < n - 1$.
- If $V_i \geq V_{max}$, then m_{n-1} (i.e., if collapsed, then restore it).

In practice, the states of cells are represented by a number between 0 and $n - 1$ (n = number of different states). A cell in state 0 corresponds to a quiescent state, while a cell in state $n - 1$ corresponds to a collapsed state. All states in between exhibit a degree of depolarization, corresponding to the number of their state. The closer the cell's state value gets to $n - 1$, then the more depolarized it becomes.

The global transition function F is defined by three rules simultaneously applied to each cell, selected according to its current state: quiescent, depolarized, or collapsed. The rules are as follows:

- If *quiescent:* It may or may not become depolarized at the next tick of the clock (i.e., $t + 1$). This depends upon the number of quiescent cells Q_{cells} in its neighborhood (normally eight neighbors), the number of collapsed cells C_{cells} in its neighborhood, and the resistance of the system to the depolarization of its cells ($R1$ and $R2$):

$$m^{t+1}(a) = \text{int}(Q_{cells}/R1) + \text{int}(C_{cells} / R2$$

- If *depolarized:* The tendency is to become more depolarized as the clock t evolves. Its state at the next tick of the clock $t + 1$ depends upon two factors: the capacitance k of the nerve cell and the degree of depolarization of its neighborhood. The degree of depolarization of the neighborhood is the sum of the numbers that correspond to the states of the neighbors S divided by the number of quiescent neighbors Q_{cells}:

$$m^{t+1}(a) = \text{int}((S / Q_{cells}) + k)$$

- If *collapsed:* A collapsed cell at time t generates a new quiescent cell at time $t + 1$:

$$m^{t+1}(a) = m_0$$

Once implemented on a computer, the user can specify the behavior of ChaOs by setting up the following parameters:

- The number of n cell values (or colors), such that $n \geq 3$
- The resistors $R1$ and $R2$
- The capacitance k
- The dimension of the grid

6.3.3 The Synthesis Engine

The CA described above is interesting because its behavior resembles the way in which most of the natural sounds produced by acoustic instruments evolve: The automaton tends to evolve from an initial wide distribution of cell's states in the grid toward oscillatory cycles of patterns. Acoustic sounds also tend to converge from a wide distribution of their partials at the onset, to periodical oscillations. The organization principle of this CA intuitively suggests that it could be applied to control the production of a large number of sonic granules that together form a complex sound event: the overall sound would begin with a highly disorganized sequence of granules that would gradually settle into an oscillatory pattern sequence. To find an effective way to map the behavior of the CA onto the parameters of the synthesis algorithm has not, however, been a straightforward task. We have devised and tested several methods, but only a few have produced interesting sounds. We introduce below the technique that has been adopted.

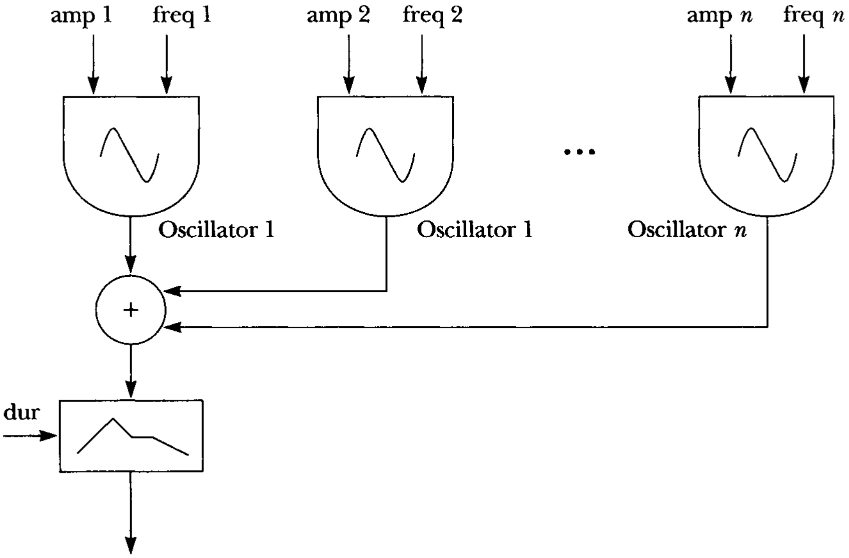
The Mapping Technique

Each sonic granule produced by the system is composed of several partials; each partial is a sine wave produced by an oscillator (in theory, we could use any other waveform here). An oscillator needs three parameters to function: frequency, amplitude, and duration (in milliseconds) of the sine wave.

The CA controls the frequency and duration values of the components of each sound granule, but the amplitude values are preset by the user beforehand; see Miranda (1998a) for a comprehensive introduction to sound synthesis techniques.

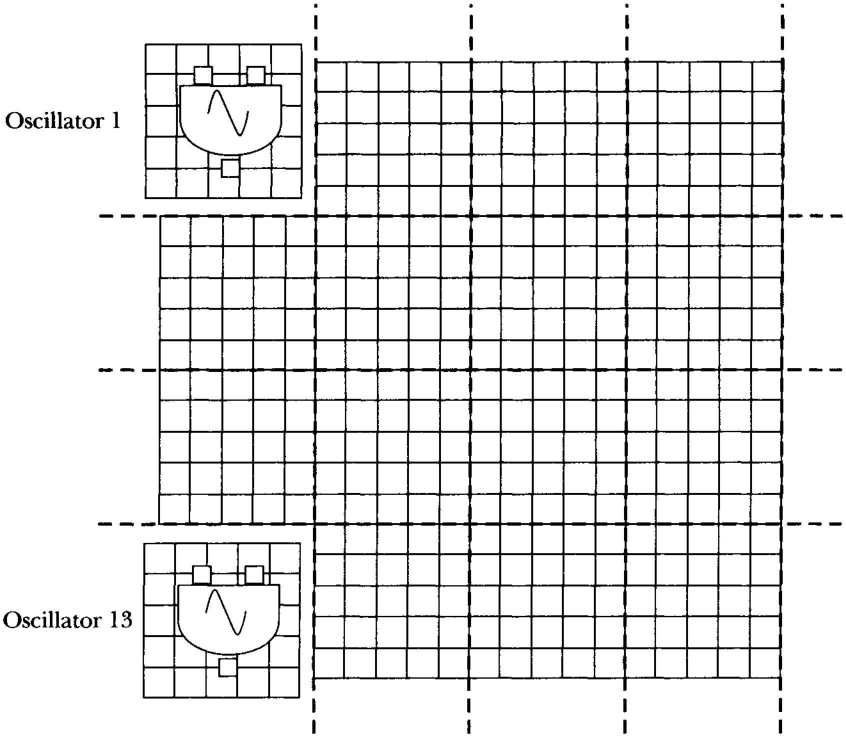
Each possible cell state m_p is associated beforehand to different frequency values (e.g., $M = \{m_0 = 110 \text{ Hz}, m_1 = 220 \text{ Hz}, m_2 = 440 \text{ Hz}, \ldots, m_n = N\}$), and oscillators are associated to a group of cells.

Each sound granule is in fact the product of the additive synthesis of sine waves (Figure 6.4): at each configuration $c, c^1, c^2, \ldots, c^n$ of cells, all oscillators simultaneously produce sine waves, whose frequencies are determined by the arithmetic mean over the frequency values associated with the states of their corresponding cells. Thus, the frequency values of the components of a granule at time t are established by the arithmetic mean of the frequencies associated with the states of the cells allocated to different oscillators. As an example, a grid of 400 cells associated with 16 oscillators of 25 cells each is shown in Figure 6.5.


amp 1
freq 1
amp 2
freq 2
amp n
freq n
...
Oscillator 1
Oscillator 1
Oscillator n
+
dur

6.4
FIGURE

The additive synthesis of sine waves.


Oscillator 1
Oscillator 13

6.5
FIGURE

An example of a grid of 400 cells allocated to 16 digital oscillators.

The duration of a whole sound event is determined by the total number n of configurations c, c^1, c^2, . . . , c^n, and the duration of the individual granules; for example, 100 configurations of granules of 40 milliseconds would result in a sound event of 4 seconds' duration.

In order to operate the system, the user specifies beforehand the dimension of the grid, the number of oscillators, the allocation of cells to oscillators, the frequencies associated with the states, the duration of the granules, and the parameters of ChaOs. These parameters are the number of states in set M, the resistors $R1$ and $R2$, the capacitance k, and the number of iterations (i.e., maximum value for t).

6.4 COMMENTARY ON THE RESULTS

The mapping method described above is interesting because it explores the behavior of ChaOs in order to produce sounds in a way that resembles the evolution of sounds produced by most acoustic instruments during their production; their harmonics converge from a wide distribution (as in the noise attack of a sound) to oscillatory patterns (the characteristic of a sustained tone). The random initialization of states in the grid produces an initial wide distribution of frequency values, which tend to settle to a periodic fluctuation.

We have synthesized sounds using up to 64 different states (that is up to 64 different frequency values) and up to 64 oscillators, on grids of up to 4,000,000 cells (2,000 × 2,000). The results have tended to exhibit a great sense of natural movement and flow, and yet most of these sounds cannot be found in the "real" acoustic world. Some results, however, resemble the sounds of flowing water, bird calls, and insects.

Variations in tone color can be achieved by varying the frequency values associated with the states of the cells. For example, if the set of frequencies contains values lower than 220 Hz, then the result will be a dull sound in the lower band of the audible range. Conversely, if the set contains frequencies above 880 Hz, then the results will be a brighter sound in the higher band of the audible range. The size of the grid and the number of cells per oscillator are largely responsible for the degree of granularity of the spectral variation: A larger number of cells per oscillator produces finer-granulation effects, while a smaller number produces coarser-granulation effects. The length of the individual granules also plays a key role in the overall result. The acoustic effect of the variation of the length of the individual granules can be summarized as follows: Very short lengths (e.g., 35 milliseconds) produce textures of sparkling bubblelike cloud of sounds, whereas larger lengths (e.g., 800 milliseconds) produce sequences of sound "strokes"; lengths above 1 second produce sequences of notes of different timbres. Different rates of transition from noise to oscillatory patterns are obtained by changing the values of $R1$, $R2$, and k.

We have used this system to generate the sounds to compose "Olivine Trees," a prize-winning electroacoustic composition (Luigi Russolo competition 1998, Italy), and "Electroacoustic Samba X," recently released on CD (Miranda 1998b).

6.5 CONCLUSIONS

The synthesizer we have described here is a truly A-Life musical instrument in the sense that it takes advantage of the behavior of cellular automata in order to produce organic synthetic sounds. For this reason it is one of the instruments of choice for our musical evolution experiments. We are currently using this instrument in a number of experiments in the context of the self-organization mechanism discussed in Section 6.2.3. In this case, the virtual agents use granular synthesis instead of percussion instruments. The resulting repertoire of sounds that emerge from these interactions is very interesting from an aesthetic point of view, but we still need to devise a set of scientific criteria to better evaluate the meaning of these results in the context of our long-term research goals.

ACKNOWLEDGMENTS

The work on the CA-based synthesiser started in 1994 as a TRACS research project at Edinburgh Parallel Computing Centre (EPCC) and at the Faculty of Music of Edinburgh University. The author is grateful to Peter Nelson (Faculty of Music), and Robert Fletcher and Martin Westhead (EPCC), for their support and valuable contributions during the early design and engineering stages.

A preliminary working version of this system for popular computer platforms exists under the name of Chaosynth. It was redesigned and programmed in collaboration with Joe Wright and is available on the accompanying CD-ROM of this author's book *Computer Sound Synthesis for the Electronic Musician* (Miranda 1998a). A fully fledged version for professional use by musicians and composers is available via Nýr Sound at *www.nyrsound.com.*

This author is grateful to Sony CSL, Paris, for supporting his current research on the origins and evolution of music.

REFERENCES

Bamberger, J. (1991). *The Mind Behind the Musical Ear: How Children Develop Musical Intelligence.* Harvard University Press.

Campbell, M., and C. Greated (1987). *The Musician's Guide to Acoustics.* Dent & Sons Ltd.

Cood, E. F. (1968). *Cellular Automata.* Academic Press.

de Boer, B. (1997). Generating Vowel Systems in a Population of Agents. In P. Husbands and I. Harvey (eds.), *Proceedings of the Fourth European Conference on Artificial Life,* MIT Press.

Degazio, B. (1997). The Evolution of Musical Organisms. *Leaonardo Music Journal* 7:27–33.

Dewdney, A. K. (1988). Computer Recreations: The Hodgepodge Machine Makes Waves. *Scientific American* (August):104–107.

Ermentrout, G. B., and L. Edelstein-Keshet (1993). Cellular Automata Approaches to Biological Modeling. *Journal of Theoretical Biology* 160.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution.* Wiley.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization & Machine Learning.* Addison-Wesley.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Koza, J. (1992). *Genetic Programming: On the Programming of Machines by Means of Natural Selection.* MIT Press.

McAlpine, K., E. R. Miranda, and S. Hoggar (1999). Composing Music with Algorithms: A Case Study System. *Computer Music Journal* 23(2).

Miranda, E. R. (1993). Cellular Automata Music: An Interdisciplinary Project. *Interface: Journal of New Music Research* 22(1):3–21.

Miranda, E. R. (1994). Music Composition Using Cellular Automata. *Languages of Design* 2.

Miranda, E. R. (1998a). *Computer Sound Synthesis for the Electronic Musician.* Focal Press.

Miranda, E. R. (1998b). Electroacoustic Samba X. *Electroacoustic Music from Latin America.* CD OO45, O.O. Discs.

Miranda, E. R. (1999). Modelling the Fundamental Mechanisms of Musical Evolution: An Artificial Life Approach. In *Proceedings of the International Computer Music Conference,* Beijing.

Reck, D. (1997). *Music of the Whole Earth.* Da Capo Press.

Roads, C. (1991). Asynchronous Granular Synthesis. In G. De Poli et al. (eds.), *Representation of Musical Signals,* MIT Press.

Schwefel, H.-P. (1965). *Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik.* Diplomarbeit, Technische Universität Berlin.

Steels, L. (1997). The Synthetic Modeling of Language Origins. *Evolution of Communication Journal* 1(1).

Steels, L., and R. Brooks (eds.) (1995). *The Artificial Life Route to Artificial Intelligence: Building Embodied Situated Agents.* Lawrence Erlbaum Associates.

Steels, L., and P. Vogt (1997). Grounding Adaptive Language Games in Robotic Agents. In P. Husbands and I. Harvey (eds.), *Fourth European Conference on Artificial Life,* MIT Press.

Truax, B. (1988). Real-Time Granular Synthesis with a DSP Computer. *Computer Music Journal* 12(2):14–26.

Washabaugh, W. (1995). The Politics of Passion: Flamenco, Power, and the Body. *Journal of Musicological Research* 15:85–112.

Wittgenstein, L. (1963). *Philosophical Investigations.* Basil Blackwell.

Wolfram, S. (1994). *Cellular Automata and Complexity.* Addison-Wesley.

Xenakis, I. (1971). *Formalized Music.* Indiana University Press.

7

CHAPTER

Vox Populi: Evolutionary Computation for Music Evolution

Artemis Moroni Technological Center for Informatics
Jonatas Manzolli University of Campinas
Fernando Von Zuben University of Campinas
Ricardo Gudwin University of Campinas

As I cannot come to you at present, I am in the meantime addressing you using a most excellent method of finding an unknown melody, recently given to us by God and I found it most useful in practice. Further, I most reverently salute Dom Martin, the Prior of the Holy Congregation, our greatest helper, and with the most earnest entreaties I commend my miserable self to his prayers, and I admonish Brother Peter, who, nourished by our milk, now feeds on the rudest barley, and after golden bowls of wine, drinks a mixture of vinegar, to remember one who remembers him.

. . .

To find an unknown melody, most blessed brother, the first and common procedure is this. You sound on the monochord the letters belonging to each neume, and by listening you will be able to learn the melody as if from hearing it sung by a teacher. But this procedure is childish, good indeed for beginners, but very bad for pupils who have made some progress. For I have seen many keen witted philosophers who had sought out not merely Italian, but French, German, and even Greek teachers for the study of this art, but who, because they relied on this procedure alone, could never become, I will not say skilled musicians, but even choristers, nor could they duplicate the performance of our choir boys.

—*Guido d'Arezzo (Strunk 1950)*

7.1 INTRODUCTION

Systems for algorithmic composition evolved side by side with the arising of Western music. One of the first known proposals to formalize composition was made by the Italian monk Guido d'Arezzo, in 1026, who resorted to using a number of simple rules that mapped liturgical texts in Gregorian[1] chant, due to an overwhelming number of orders for his compositions. In the classical era, composers such as Mozart, Haydn, and C. P. E. Bach used an algorithmic decision process called "Würfelspiel" (Dice Game) to compose minuets and other works. The music was constructed by means of random selection of segments from a table of motifs. Not surprisingly, in the 20th century, John Cage used Tarot and I Ching to build musical architectures (Loy 1988).

The development of electrical and electronic devices brought electronic musical instruments to the musical realm. Nowadays, the computer represents a technological tool to study music in a way that was not possible in the past. As stated by Moore (1990):

> Computers allow precise, repeatable experimentation with sound. In effect, musicians can now design sounds according to the needs of their music rather than relying on a relatively small number of traditional instruments . . . New devices extend the capabilities of musicians to control the production of sound during live performances. Whatever else they may represent, computers are the most flexible and most powerful instruments of music yet devised.

Several approaches to composing music with the aid of a computer have been developed since Max Mathews's early research (1963), which brought to sonic composition the computer as a new musical instrument, or the composition of the ILLIAC suite by means of stochastic processes (Hiller and Baker 1964). In the last four decades, much research has brought about many computer music systems, with models derived from nonlinear dynamics (Bidlack 1992; Manzolli 1993). Other methods have explored formal grammars and their extensions, sometimes in the direction of cellular automata (Miranda 1994, and Chapter 6).

---

[1] The designation "Gregorian" refers to Pope Gregory I, who ruled from 590 to 604, and who is generally believed to have played a decisive role in the final arrangement of the chants, each of which he (or rather those to whom he had entrusted the task) assigned to a specific occasion of the liturgical year, according to a broadly conceived plan (Apel 1958).

Musical composition can be seen as a framework to express human subjectivity in symbolic and subsymbolic terms. Such a field is exactly the type of domain in which an interactive genetic algorithm (IGA) is most useful. An application of genetic algorithms to generate jazz solos has been described in Chapter 5, and this technique has also been studied as a way of controlling rhythmic structures (Horovitz 1994). While recent techniques of digital sound synthesis bring a large number of new sounds to the musician's desktop, several artificial intelligence techniques have been applied to algorithmic composition. On the one hand, a new sound imagery; on the other hand, several approaches attempt to organize a wide panoply of new sounds:

> The general area of compositional algorithms reflects the tremendous variation in approach to music making which has been the hallmark of contemporary music for decades (Moore 1990).

In this chapter we introduce a new system, Vox Populi, based on evolutionary computation, for composing music in real time. A population of chords is properly codified according to the MIDI protocol and evolves by the application of genetic algorithms. A fitness criterion is defined to indicate the best chord in each generation, and this chord is selected as the next element in the sequence to be played. Each new generated chord is a new sound palette that a musician can use to continue the music evolution. Graphic controls (pad and sliders) provide user-friendly manipulation of the fitness and of the sound attributes. Evolutionary computation is used to stimulate the user with novel sounds, and it allows the user to respond.

Associating the dynamic behavior of genetic algorithms with these tools for real-time interaction, Vox Populi becomes a musical instrument. But unlike a traditional instrument, Vox Populi is able to create its own sound raw material (chord population) and to provide choice criteria (music fitness) simultaneously. All these features enhance the user's music capabilities and mark this system as the state of the art in computer music.

Next, a general description of the main components of the computational environment and melodic, harmonic, and voice range criteria for musical fitness are defined. Section 7.2 introduces the auditory attributes that are considered in this application. Section 7.3 explains the genetic encoding of notes and the evolutionary cycle for chord production. Section 7.4 describes the musical criteria applied as fitness functions. In Section 7.5 the graphical interface is presented, and the control bars are depicted. Section 7.6 discusses the experiments, and Section 7.7 presents the conclusions.

7.2 SOUND ATTRIBUTES

For our purposes, a sound, or an auditory event, is characterized by four parameters: pitch, timbre, loudness, and duration.

Pitch can be defined as the auditory property of a note that is conditioned by its frequency relative to the other notes. The range of musical pitch has been defined as the range within which the interval of an octave can be perceived. This has been found to correspond roughly to the range of the piano. From this continuum of frequencies, a set of discrete frequencies is selected in such a way that the frequencies bear a definite interval relationship among them. So, pitch in the musical sense corresponds to a frequency that is selected from a predefined repertoire. In this scheme, two discrete frequencies are chosen in the interval of an octave, so that the ratio between any two adjacent frequencies is $2^{1/12}$. In music terminology, this interval ratio is termed a *semitone.*

The other three parameters can be described straightforwardly: *timbre* is the individuality of sound acquired by the addition of harmonics to the fundamental pitch and is characteristic of a given musical instrument and the way of playing it. *Loudness* is the aspect of an auditory event related to its intensity, and, finally, *duration* is characterized by the period of time during which the event is perceivable.

Using these concepts, a *melody* is defined as a fixed temporal ordering of auditory events. In conventional occidental notation, a melody resembles a system of Cartesian coordinates. The pitch and duration are carefully marked; timbre is decided by the instrument for which it is written, and loudness is marked more crudely (Vidyamurthy and Chakrapani 1992).

The MIDI protocol provides a symbolic representation for the musical notes from which the above attributes may be extracted. Our approach uses MIDI events described by the MIDI note table for pitch representation, the MIDI velocity table for loudness and the general MIDI table for timbre. The duration is set in milliseconds.

7.3 EVOLUTIONARY MUSICAL CYCLE

In our approach, the population is made up of groups of four notes, which are potential solutions for a selection process. Genetic algorithms are used to generate and evaluate a sequence of chords. This sequence produces a sound result resembling a chord cadence or a fast counterpoint of note blocks. These are sent

1011111	1010111	0010111	0100111

7.1 FIGURE The structure of a MIDI chromosome.

to the MIDI port and can be heard as sound events in real time. Melodic, harmonic, and voice range fitnesses are used to control musical features. Based on the ordering of consonance of musical intervals, the concept of approximating a sequence of notes to its harmonically compatible note or tonal center is used. This method employs fuzzy formalism and is posed as an optimization approach based on factors relevant to hearing music, further described in Section 7.4.

7.3.1 The Voices Population

Individuals of the population are defined as groups of four voices, or notes. From now on, "voices" and "notes" will be used interchangeably. These voices are randomly generated in the interval [0..127], which corresponds to 7-bit values for a MIDI event. In each generation, 30 groups are generated. The group, or chord, is internally represented as a chromosome of 28 bits, or 4 words of 7 bits, one for each voice (Figure 7.1).

The following steps make up the genetic cycle (Pedrycz and Gomide 1998):

1. Create an initial population randomly.
2. While not stopped (by the user), perform the following:
   - Evaluate the musical fitness of each individual in the population.
   - Apply the genetic operators to the population of MIDI chromosomes (groups of voices), chosen based on musical fitness, to create a new population. The operators are *reproduction* (copy existing individual strings for a new population); *crossover* (create two new chromosomes by crossing over randomly chosen sublists (substrings) from two existing chromosomes); *mutation* (create new chromosomes from an existing one by randomly mutating a character in the list).
3. Find the best individual in the new population and play it as a MIDI event. Go to step 2.

The general architecture of the genetic cycle is depicted in Figure 7.2.

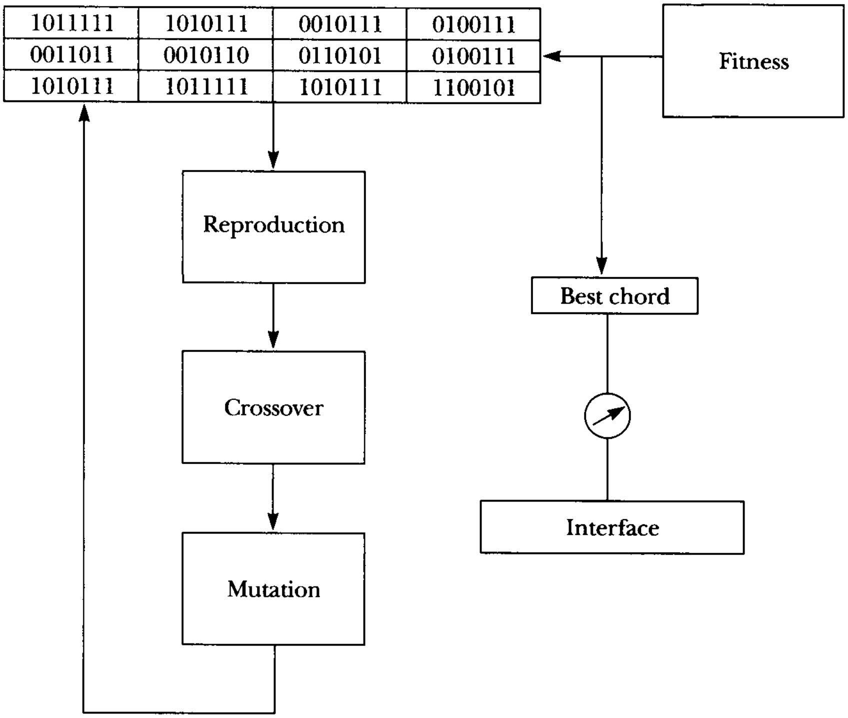

1011111
1010111
0010111
0100111
0011011
0010110
0110101
0100111
1010111
1011111
1010111
1100101
Fitness
Reproduction
Best chord
Crossover
Interface
Mutation

FIGURE 7.2 The genetic cycle.

7.3.2 The Rhythm of the Evolution

Musicians have always found inspiration in nature for their compositions. Water, fire, wind, rain, birds, and other natural sounds have appeared in their work. Prokofiev, in *Peter and the Wolf,* suggests both the sound of men and other animals. Earlier, *The Magic Flute,* by Mozart, explored the sounds of birds. Electroacoustic music uses the soundscape, quotidian noise, automobiles, hinges, and coins, in compositional context. Villa-Lobos explored several themes inspired by the Brazilian landscape. In *Bachiana No. 4,* he explored a train traveling along the countryside as a theme for developing his music. Sounds from the old Brazilian railroad were used by Raul do Valle and Jônatas Manzolli in a multimedia performance named "Trilhos Sonoros da Ferrovia" (Railroad Track Sound). A few years ago, in São Paulo, a group called The Sound Hunters roamed about the city recording the sounds of traffic, fountains, rush hour, and other everyday sounds, using them as samples in their compositions.

In a similar manner, inspired by machines with cyclical movements, such as trains or old mechanical sewing machines, we searched for a machine rhythm to express the Vox Populi evolution. What would be the rhythmic pattern of a computer when it is running a genetic algorithm? We decided to generate music with a rhythm built by the time necessary for selecting a better group of voices.

Two processes were integrated: (1) an evolving process generating individuals—groups of notes or voices, in this case—applying genetic operators and selecting individuals, and (2) the interface to look for notes to be played by the computer. When a best group is selected, it is put in a critical region that is continuously verified by the interface. These notes are played until the next group is selected. The timing of these two processes determines the rhythm of the music that is being heard. In any case, a graphical interface allows the user to interact with the rhythm, by modifying the cycles.

The steps above illustrate how many operations are executed in each cycle. The time interval between the selection of the best chords in two successive cycles may be different, while the interface is regularly "asking for new notes." Although the time taken to designate the best individual in each generation is approximately the same, small variations in each time cycle determine the *genetic rhythm;* the different times for the notes being played are perceived as the rhythm of the melody generated by the genetic cycle. The resulting system, Vox Populi, allows the user to modify the fitness function by means of four controls: The first is the melodic criterion; the second, the duration of the genetic cycle and musical rhythm; the third is the set of octave ranges to be considered; and the fourth, the time segment for each selected orchestra. All these controls are available for real-time performance, allowing the user to play and interact with Vox Populi's music evolution.

7.4 FITNESS EVALUATION

Since Western music is based on harmony, any general theory of music must address this matter. The term "harmony" is inherently ambiguous: It refers both to a lower level, where smoothness and roughness are evaluated, and to a higher aesthetic level, where harmony is functional in a given style. However, harmony is very subjective; the judgment of harmony does not seem to have a natural basis, but appears to be a common response acquired by people in a certain cultural setting. Therefore, opinions on the subject may vary widely depending on social and cultural backgrounds, and the many attempts to formalize the concept have proven inadequate. Nevertheless, while there is a difference of opinion on what constitutes harmony, there is a general agreement on the relative order of music interval consonance. Numerological theories of consonance have attempted to capture this aspect, but here again, a lot is left to the imagination, as the theory does not clearly delineate what constitutes the order of simplicity of music intervals.

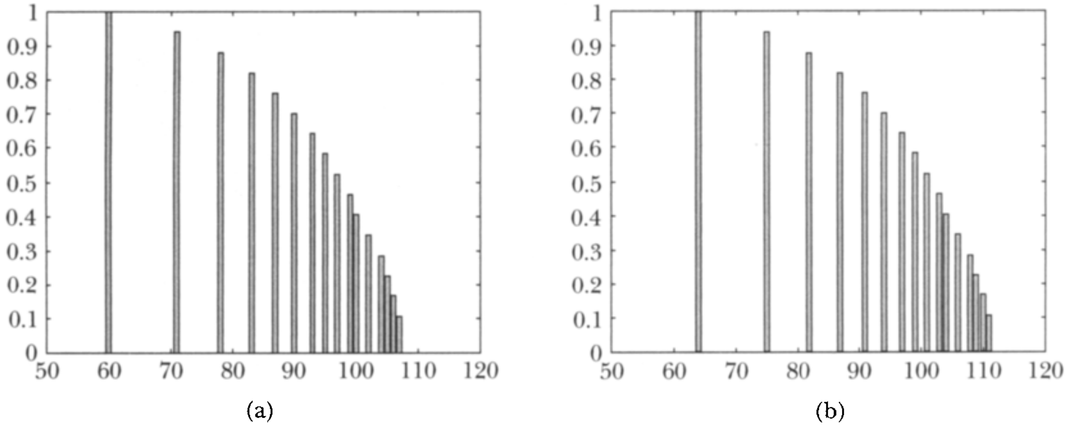
In our case, as a fitness function, a numerological theory of consonance from a physical point of view, based on a relative ordering of musical intervals, has been applied. Based on the ordering of consonance of these intervals, a sequence of notes is approximated to its harmonically compatible note or tonal center. Tonal centers can be thought of as an approximation of the melody that describes its flow. This method uses fuzzy formalism and is posed as an optimization problem based on physiological factors relevant to listening to music. This approach is significant because it does not adopt any heuristics.

7.4.1 The Consonance Criterion

Consonance is a combination of two simultaneous notes judged to be pleasing to the ear. In Vidyamurthy and Chakrapani (1992), harmony is introduced as a function of the commonality or the overlap of the harmonic series components of given fundamental notes. This overlap measurement is then scaled to a value between 0 and 1, with 1 denoting complete overlap (i.e., the two notes are the same) and 0 denoting no overlap at all. This concept of overlap can be succinctly captured in a fuzzy-set-based formalism.

A musical note is a compound tone consisting of its primary tone and upper harmonic series tones. It can be associated with a fuzzy set in which the degree of membership of each tone is proportional to its amplitude. Represented graphically as a spectrum, a musical note is a graph of frequency versus amplitude. To further enhance the musical comprehensibility of a note, the audible range frequency is mapped onto the keys of a piano. Using the chromatic scale, it is possible to make an approximation of this model using the piano keyboard. In what follows, the weighting of the note partials versus the harmonic series is presented, where n denotes the nth key on the piano, and $(n + k)$ denotes the key k semitones above key n. In Figure 7.3 the harmonic series for notes 60 and 64 are shown.

Formally, each note N is a fuzzy set S_N, comprising a collection of pairs (x, y), where x is a tone (also called a "partial"), and y is its "weight" in the note, which corresponds with the amplitude of that tone when the note is played. For example, the set S_{60} contains 16 such pairs and is illustrated by the graph in Figure 7.3 (a), which plots the points (x, y). When we thus represent a note as a fuzzy set, the y values are normalized so that the total weights of all partials sum to 1.



7.3 FIGURE The upper partials for two notes: (a) 60 and (b) 64.

If S_N is a note, and U is the set of all tones, we therefore have

$$S_N = \{(x,y) \mid x \in U, y \in [0,1]\}, \text{ where } \sum_{y \in \{y \mid (x,y) \in S_N\}} y = 1$$

Given a pair of notes S_M, S_N, we can define their *intersection* $S_{M \cap N}$, as follows:

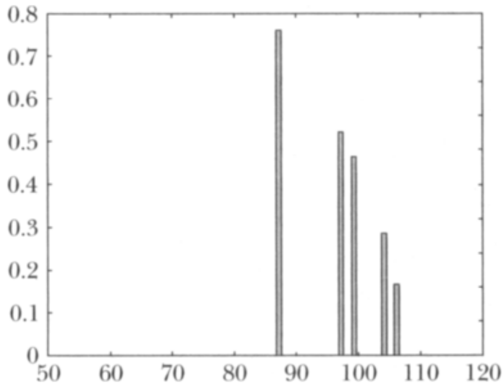
$$S_{M \cap N} = \{(x,y) \mid (x,a) \in S_M \wedge (x,b) \in S_N \wedge y = \min\{a,b\}\}$$

That is, the intersection is a collection of partials involving only the tones represented in both notes, and with each such tone given the lowest of the weights it has in the notes themselves. For example, Figure 7.4 depicts the intersection of the two notes shown in Figure 7.3.

We can now define the consonance, or overlap, of a pair of notes S_M, S_N, as follows:

$$C_O(S_M, S_N) = \sum_{(x,y) \in S_{M \cap N}} y$$

Simply stated, the consonance of a pair of notes may be interpreted as the sum of the intersection of the partial weights, with results in the range [0,1].



7.4 FIGURE The weighting of the *intersection* of notes 60 and 64.

7.4.2 Melodic Fitness

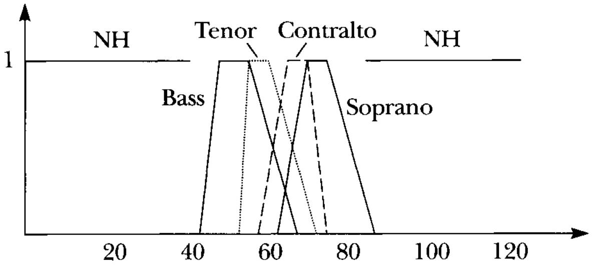
Melody is defined as an ordered sequence of notes with their corresponding start times. In formal terms, a melody can be a string in which each character is an ordered pair (S_N, t), where S_N is the note and t is its duration. Given an ordered sequence of notes, it seems intuitively appealing to call the note that is most consonant with all the other notes, the "coloring" or "tonal center." Hence, the extraction of the tonal center of a sequence of notes would involve finding a single harmonically compatible note, such that the time weighted dissonance between a given note and any other one in the sequence is minimized. In Vox Populi, this is measured according to the weighting value *Id,* which is obtained from the interface control and can be varied by the user. To derive the melodic fitness of a string of four notes $(x_1,t_1),(x_2,t_2),(x_3,t_3),(x_4,t_4)$, we therefore first derive their tonal center note *Id* and calculate

$$M(x_1,x_2,x_3,x_4) = \max_{i=1,\dots,4}\{Co(x_i,Id)\}$$

7.4.3 Harmonic Fitness

Harmony is defined here as a function of the commonality or overlap of the spectral components of the notes and the sum of the two-by-two consonance. Therefore, the harmonic fitness is defined as

$$H(x_1,x_2,x_3,x_4) = Co(S_{x_1},S_{x_2}) + Co(S_{x_2},S_{x_3}) + Co(S_{x_3},S_{x_4}) + \mathrm{Co}(S_{x_1},S_{x_4})$$



7.5 FIGURE The linguistic values associated with the voices.

7.4.4 Voice Range Criterion

Voices are associated with the linguistic terms "bass," "tenor," "contralto," "soprano," and "nonhuman" (NH). The related fuzzy sets are shown in Figure 7.5, where the units on the horizontal axis are MIDI note values. Each voice is assigned to a membership value associated with each linguistic term in the set {NH, B, T, C, S}. For example, the MIDI note value 60 has the degrees of membership 0.5, 1.0, and 0.5 in the bass, tenor, and contralto sets, respectively. Given a note, the membership of that note in each set is calculated, and the maximum value is taken to indicate the voice of that note. In case of a tie, the voice classification is based on the distance of that note from the centers of the two fuzzy sets concerned.

Once the voices of each group are evaluated according to their membership in the interval of voices, the voice range criterion returns the maximum in each interval. Therefore, the voice range fitness is evaluated as

$$O(x_1, x_2, x_3, x_4) = \left(\sum_{i \in \{1,\ldots,4\}} V(x_i) \right) / 4$$

where $V(x_i)$ is the membership value of note x_i associated with its assigned voice.

7.4.5 Musical Fitness

The resulting musical fitness is a conjunction of the previous functions and is defined as

$$F(O,M,H) = O(x_1,x_2,x_3,x_4) + M(x_1,x_2,x_3,x_4) + H(x_1,x_2,x_3,x_4)$$

In the selection process, the group of voices with the highest fitness is selected and played.

7.5 INTERFACE AND PARAMETER CONTROL

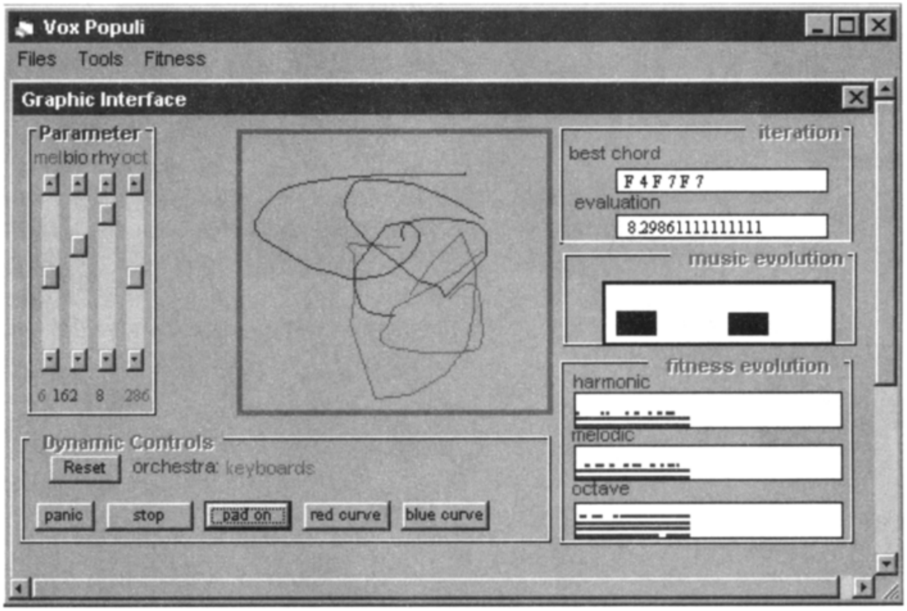
The system was designed to perform a series of sound experiments and to be used as a tool for algorithmic composition, with satisfactory performance, the ability to interact with sound, and a possible use of Vox Populi for live music. The program was developed for the MS Windows environment and runs on any personal computer and most commercial soundboards. The graphical interface allows the user to change parameters and interfere with the fitness function and, consequently, with the musical evolution.

The resulting music moves from very pontilistic sounds to sustained chords; it depends on the duration of the genetic cycle and the number of individuals in the original population. The octave fitness forces the notes to be in the range *H,* assumed to be the range reached by the human voice and associated with the center region of the notes on the piano, but since several orchestras of instruments are used, this range is too limited for some of them. The original decision to restrict the generated voices to specific ranges was only to resemble the voices to human voices; nevertheless, a user can enlarge these ranges by using the octave control. The interface was designed to be flexible so that the user can modify the music being generated.

Next we present a short description of the controls available for user interaction with Vox Populi. The descriptions should be read with reference to Figure 7.6, which shows Vox Populi's graphical interface.

Melodic control: The Mel scroll bar allows the user to modify the value of *Id,* the tonal center in the evaluation of the melodic fitness.

Biological control: The Bio scroll bar allows the user to interfere with the duration of the genetic cycle, modifying the time between genetic iterations. Since the music is being generated in real time, this artifice was necessary for the timing of the different processes that are running. This value determines the slice of time necessary to apply the genetic operators, such as crossover and mutation, and can also be interpreted as the reproduction time for each generation.


Vox Populi
Files Tools Fitness
Graphic Interface
Parameter
mel bio rhy oct
6 162 8 286
iteration
music evolution
fitness evolution
harmonic
melodic
octave
Dynamic Controls
Reset
panic
stop
pad on
red curve
blue curve

best chord F 4 F 7 F 7

evaluation 8.29861111111111

orchestra: keyboards

7.6 FIGURE Vox Populi's graphical interface.

Rhythmic control: The Rhy scroll bar changes the time between evaluations of the musical fitness. It determines the "time to produce a new generation," or the slice of time necessary to evaluate the musical fitness of the population. It interferes directly with the musical rhythm. Changes in this control make the rhythm faster or slower.

Octave control: The Oct scroll bar allows the interval of voices considered in the voice range criterion to be enlarged or decreased.

Orchestra control: Six MIDI orchestras are used to play the selected chords: (1) keyboards; (2) strings and brasses; (3) keyboards, strings, and percussion; (4) percussion; (5) sound effects; and (6) a random orchestra, by taking an instrument from the general MIDI list. Using the order above, these orchestras are sequentially changed in time segments controlled by the Seg scroll bar.

Interactive pad control: The Pad On button enables and disables the pad's effect on the controls defined above. These controls can be coupled in two pairs, which can be interpreted as variables of a two-dimensional phase space, allowing a user to draw an oriented curve to determine the musical evolution. There are two curves, each associated with a different color. The red curve describes a phase space of the *melodic* and *octave range* control variables. The blue curve describes a phase space of the *biological* and *rhythmic* control variables.

The pad may be musically interpreted as an elementary tool that allows a "master gesture" to conduct the music.

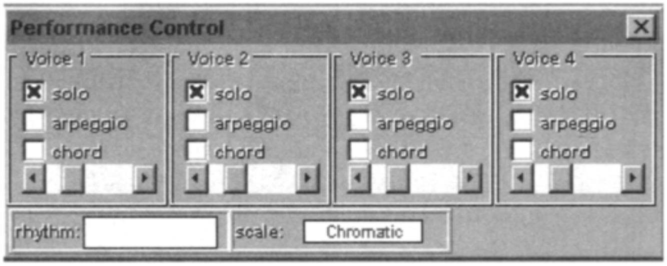
Fitness displays: Three other displays allow the user to follow the fitness evolution. The first display, on the top right, shows the notes and the fitness of the chord that is being played. In the middle, a bar graph shows the four voices (bass, tenor, contralto, soprano) and their value. It is equivalent to the membership function values related to the range of the voices. The last display shows the melodic, harmonic, and octave fitness bars.

7.6 EXPERIMENTS

Two main approaches were tried to express the fitness evaluation, and both presented interesting results. The first one, derived from a heuristic model, provides a less expensive fitness evaluation. The disadvantage of this approach is that it demands a strong musical background to formulate and understand it. Because of this, its presentation will be omitted. This method allows the use of a large population, between 100 and 200 chords, producing more diversification and resulting in a slow convergence to the best chord sequence.

In the second approach, the consonance criterion presented above is used, and a longer calculation is needed to evaluate the music fitness. In order to assure good real-time performance, the population is limited to 30 chords. Since the musical fitness utilized was tighter, the resulting sound output was less diversified, and it was possible to hear the musical sequence converging to unison. This fact highlights the idea that in musical composition not only is the consonance desirable, but also the dissonance.

The octave control offers an interesting tool to vary the voice performance, diminishing or enlarging the voice range. The resulting sounds are isolated note blocks or chords. The larger the range, the larger the chord separation. This control highlights the automated nature of the Vox Populi very well since it produces some combinations of notes that cannot be played by a human musician because of physical limitations (distance among the notes and speed, for example). Another feature that has been added to the system is a music performance menu (Figure 7.7), which offers the possibility of playing solos, chords, or arpeggios.



Performance Control

Voice 1
☒ solo
☐ arpeggio
☐ chord

Voice 2
☒ solo
☐ arpeggio
☐ chord

Voice 3
☒ solo
☐ arpeggio
☐ chord

Voice 4
☒ solo
☐ arpeggio
☐ chord

rhythm:
scale: Chromatic

7.7 FIGURE Performance menu.

Using this entire range of musical features, impromptu musical sessions were held with live musicians, one controlling Vox Populi and the other accompanying him with a synthesizer keyboard. As a musical partner, Vox Populi can produce interesting rhythmic patterns, chord cadences, or pontilistic sound sequences for long periods. Sometimes the music seemed slightly predictable, but when the pad control was used to vary Vox Populi's parameters, the music changed to unexpected cadences, then converged to another pointillistic sequence, and so on. Maybe the best expression to describe this dialogue between the system and live musicians would be "spontaneity."

7.7 CONCLUSIONS

Vox Populi uses the computer and the mouse as real-time music controllers, producing dynamic musical structures based on evolutionary models. It is a new interactive computer-based musical instrument. It explores evolutionary computation in the context of algorithmic composition and uses a graphical interface to change the musical evolution. These results reflect current concerns at the forefront of computer music: interactive composition and the development of new controller interfaces. A demo of Vox Populi is available on the CD-ROM and at *http://www.ia.cti.br/~artemis/voxpopuli.*

The use of Vox Populi as a compositional tool in a real-time algorithmic composition environment at the electronic music studio of NICS (Interdisciplinary Nucleus of Sound Communication) showed the potential of this system to control the evolution of music. As a representation of a musical creative process, this system was integrated to interpretative improvisations created by musicians producing pleasant musical cadences. This interactivity emphasizes aspects of musical practices in the scope of human/machine interaction.

Furthermore, we have been able to integrate this system with gesture interfaces, such as gloves, to enhance the human/machine interaction, with the goal of allowing a human gesture to be the real-time controller. This is a natural extension of the pad control. This approach was previously applied to the design of ActContAct (Manzolli, Moroni, and Matallo 1998), where an electronic shoe was used for music generation. In this way, Vox Populi, designed to reflect the natural evolution of the sound domain, will return its control to the human agent during its own life cycle. It would certainly be interesting to follow up on the use of computers to develop new ways of human interaction through music.

ACKNOWLEDGMENTS

We would like to thank our fellow student Leonardo N. S. Pereira for developing the routines to evaluate the consonance criterion. This work was supported by FAPESP (São Paulo State Research Foundation) and CTI (Technological Center for Informatics), and CNPq (process no. 300910/96–7).

REFERENCES

Apel, W. (1958). *Gregorian Chant.* Indiana University Press.

Bidlack, R. (1992). Chaotic Systems as Simple (but Complex) Compositional Algorithms. *Computer Music Journal* 16(3):33–42.

Hiller, L., and R. Baker (1964). Computer Cantata: A Study of Compositional Method. *Perspectives of New Music* 3(1).

Horovitz, D. (1994). Generating Rhythms with Genetic Algorithms. In *Proceedings of the International Computer Music Conference* (*ICMC '94*), pp. 142–143.

Loy, G. (1988). Composing with Computers—A Survey of Some Compositional Formalism and Music Programming Languages. In M. V. Matthews and J. R. Pierce (eds.), *Current Directions in Computer Music Research,* MIT Press, pp. 291–396.

Manzolli, J. (1993). *Non-linear Linear Dynamics and Fractals as a Model for Sound Synthesis and Real Time Composition.* Ph.D. dissertation, University of Nottingham, UK.

Manzolli, J., A. Moroni, and C. Matallo (1998). AtoConAto: New Media Performance for Video and Interactive Tap Shoes Music. In *Proceedings of the 6th ACM International Multimedia Conference.*

Mathews, M. (1963). The Digital Computer as a Musical Instrument. *Science* 142.

Miranda, E. R. (1994). Music Composition Using Cellular Automata. *Languages of Design* 2:105–117.

Moore, R. F. (1990). *Elements of Computer Music.* Prentice-Hall.

Pedrycz, W., and F. Gomide (1998). *An Introduction to Fuzzy Sets Analysis and Design.* MIT Press.

Strunk, O. (1950) *Source Readings in Music History.* Vail-Ballou Press.

Vidyamurthy, G., and J. Chakrapani (1992). Cognition of Tonal Centers: A Fuzzy Approach. *Computer Music Journal* 16(2).

This Page Intentionally Left Blank

8 CHAPTER

The Sound Gallery—An Interactive A-Life Artwork

Sam Woolf University of Sussex at Brighton
Adrian Thompson University of Sussex at Brighton

To the memories of Ian Woolf and Julia Jolly.

8.1 INTRODUCTION

The Sound Gallery is an ongoing experiment in art and science that grew out of our research in the field of evolutionary electronics at the University of Sussex, England. The aim of this experiment is to utilize ideas and techniques derived from the science of artificial life (A-Life) to create an interactive and adaptive installation artwork that displays aesthetically interesting behavior.

The field of artificial life is one that has been explored by a number of recent artists. It has spawned a diversity of new media and has proven a rich source of inspiration. There now exists a substantial body of work belonging to the canon of A-Life art, and a growing number of artist/scientists who are increasingly active in the field. The Sound Gallery adds a new artwork to this lineage, drawing inspiration from other works of A-Life art, but also making some notable advances. It achieves interactive and adaptive behavior in novel ways, making use of reconfigurable hardware technology that (as far as we are aware) has not before been appropriated for artistic ends.

What is music? The composer Arnold Schoenberg claimed that music could be defined as "repetition and variation." The aim of the Sound Gallery is to take this statement literally, and to its logical extreme: A sound source is transformed into "music" through repetition and distortion as it plays through four separate speakers simultaneously. The signal to each speaker is endowed with variations by the intervention of separate signal-processing circuits. The electronic character and makeup of these circuits, and hence the precise distortion effects they entail, are perpetually in a state of flux and change. The circuits are constantly manipulated and reconfigured through a process of artificial evolution driven by aesthetic selection pressures.

The four speakers are to be positioned in an open gallery setting. Audience members, passersby, and curious parties will be encouraged to interact with this gallery space in whatever ways they feel fit. They should allow themselves to drift freely through the gallery space, exploring the soundscape generated by the speakers within it, seeking out those areas they find the most aesthetically pleasing, or at worst, those that are the least repellent . . .

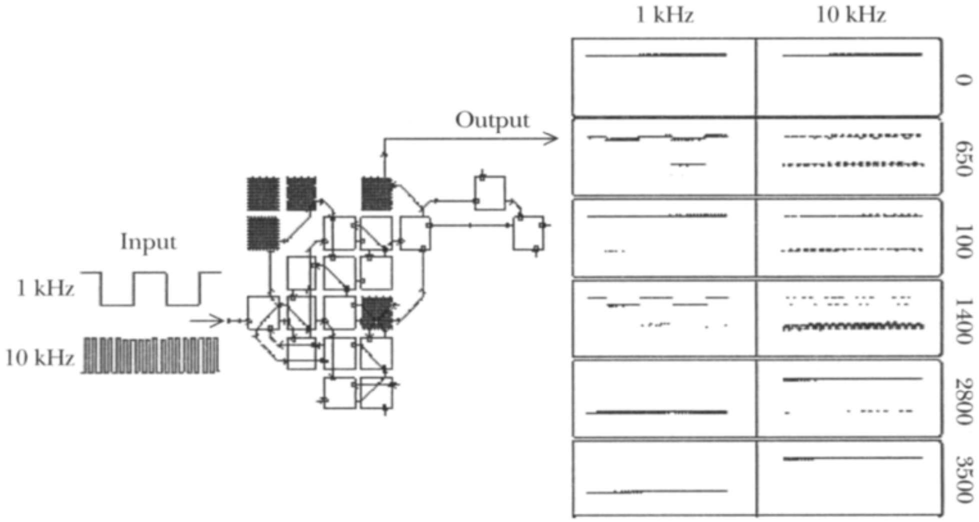
Sensors in the gallery will track the movements of those present over time. Fitness values for each of the four speakers will be derived from this information and passed to a genetic algorithm controlling the evolution of the four reconfigurable circuits. The fitness value of a particular speaker will be calculated from the movements of the audience and the ways in which individuals interact with that speaker. The fitness definition used will be one that has been designed to place the four speakers in direct competition, each one struggling against its adversaries for the possession of a scarce resource—the attention and appreciation of human spectators.

This chapter will describe and contextualize the Sound Gallery and the ideas that inform it. A working prototype of the Sound Gallery system was developed at the University of Sussex during the summer of 1999. We shall discuss the behavior and substance of this prototype model and also consider some of the refinements still to be made. But first, we speak of "evolvable hardware" and its practical and aesthetic pertinence to this project.

8.2 EVOLVABLE HARDWARE

Although many of the more mechanistic aspects of electronics design have been automated, at present an inventive skilled human must still be at the helm of the computer-aided design (CAD) tools that are now in common use. This person still experiences the pleasure of engaging in a creative activity. However, over the past five years, some researchers have begun exploring the idea of using the apparently ingenious, opportunistic, and innovative aspects of artificial evolution to aid the creativity of human electronics practitioners (Higuchi, Iwata, and Weixin 1997; Sipper, Mange, and Perez-Uribe 1998; Miller et al. 2000).

In natural evolution there is no clear long-term objective. Adaptations are made according to environmental pressures, and these change over time—especially since some of the most important parts of the environment are other life forms doing the same thing. Even when there is an obvious demand for a particular short-term improvement, such as frustrating a new predator, it is often hard to see what changes to the body or behavior would be best. Artificial evolution



8.1
FIGURE

The inspiring evolved tone discriminator circuit. Only the wiring between the components is shown, though their behaviors were also subject to evolution.

for electronics design has usually been much more constrained, with a prespecified target behavior as the only objective. Even given such a rigid goal, if the conventional rules of electronics design are not enforced, evolution can produce circuits that seem alien to a trained designer. When such ingenious but bizarre circuits are presented to a group of engineers, in our experience one half is delighted and the other is disgusted. (This can even happen with only one engineer in the group . . .)

Perhaps the first example of this is shown in Figure 8.1. The circuit has a single input and a single output and was required to discriminate between a high-pitched audio tone (10 kHz) and a lower-pitched one (1 kHz). The output should be a steady high voltage whenever one tone is present at the input, and a steady low for the other. The components shaded grey are exploited through parasitic coupling, even though they are not directly wired to the rest of the circuit. At the right of the figure, the responses of some of the circuit's ancestors are shown, starting with an initial random population (generation 0), and finishing with a near-perfect result (3,500 generations). Even though the circuit is small and the external behavior is simple, it took about two weeks' hard effort from two skilled designers to analyze roughly how it worked (Thompson and Layzell 1999).

Some of the tricks used by the circuit are so unusual that sometimes the analysts' experience of conventional design was positively unhelpful. Some of the investigative techniques needed were more akin to neuroscience than to normal electronics. It turns out that the central part of the circuit, stumbled upon by chance early in evolution, is an oscillator with two different stable modes of oscillation. Which of the two modes it falls into is very sensitively determined by the conditions on the silicon chip at that time. In fact, to discriminate between low and high tones, these sensitive dynamics are used to transduce and amplify a tiny and normally undetectable "parasitic" side effect of the way the chip is made. This mechanism was fine-tuned during the rest of evolution, and other components were added to give the high/low output rather than an oscillation. The whole is a finely balanced system, exploiting many of the precise time delays and physical properties of the silicon components. Such exploration beyond the scope of conventional design (Thompson, Layzell, and Zebulum 1999) could have exciting engineering implications, especially when steps are taken to ensure robustness over a variety of conditions of temperature, fabrication variation, and so on (Thompson and Layzell 2000).

These experiments within the field of evolvable hardware, and the strikingly unconventional qualities of the circuits they produced, set the stage for the Sound Gallery and inspired one of its key aims: to enable random and untrained members of the public to participate, perhaps unwittingly, in the design of complex and subtle electronic structures, hence demystifying and subverting the design process while simultaneously shrouding its final product with an almost indecipherable and "alien" complexity. The meandering public are to become expert electronics designers and alchemists, the sum of their base drives and motivations transformed to silicon gold.

The direct exploitation and manipulation of the physical properties and behaviors of real interconnected components embodied on a silicon chip provides more freedom to the processes of evolution than some other possible implementations of the Sound Gallery system (for example, the evolutionary "tweaking" of the parameters of a digital signal processing algorithm). It potentially allows for a far greater range of unpredictable behaviors, as evolution may exploit subtle properties of the silicon to unforeseeable and unrepeatable effect. It also renders this behavior tangible, as each evolved circuit exists as a true physical entity, and not merely as a range of parametric values stored in a computer memory buffer.

On a more practical note, the use of reconfigurable chips avoids the need to evaluate the evolving circuits in a software simulation. Although important in the developing field of evolutionary electronics, simulations of circuit behavior inevitably constrain possibilities, or miss nuances of physical reality, allowing only the smallest and most simplistic of circuits to be simulated fast enough to accommodate real-time interactive behavior.

8.2.1 Reconfigurable Chips

Reconfigurable chips consist of an array of components interspersed with wires. There are electronic switches distributed throughout that control the behaviors of the components and how they connect to the wires. In the types most useful for evolution, the electronic switches are made from transistors turned on or off according to the 1 or 0 contents of adjacent bits of an on-chip RAM memory. Any computer can be interfaced to the device, and the computer's software can write to this configuration memory just like a normal memory chip. By doing so, a particular circuit is physically instantiated in silicon by the reconfigurable switches. This circuit then behaves in real-time according to the laws of semiconductor physics, rather than as a computer program stepping through a sequence of instructions.

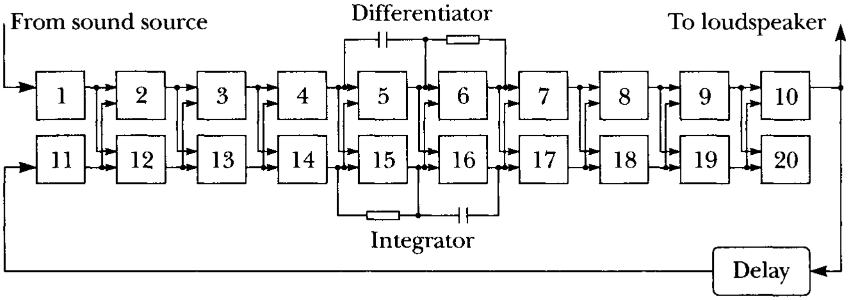
Many kinds of reconfigurable chip exist, such as field-programmable gate arrays (FPGAs) intended for digital use (Oldfield and Dorf 1995) (as abused in the tone-recognition experiment above), or arrays of analog components. The Sound Gallery uses the Zetex Totally Reconfigurable Analog Circuit (TRAC),[1] with a separate TRAC020 chip manipulating the sound on its way to each loudspeaker. The four TRAC020 chips are housed on a Zetex development board that connects to the computer via a parallel cable. The genetic algorithm software, controlling the individual circuits configured onto each chip, runs on a single PC, as will be described later.

The layout of the TRAC chip, along with its connections to some external components useful in our application, is shown in Figure 8.2. The component cells, or "configurable analog blocks" (CABs), are arranged in two rows of 10. The basic signal flow is from left to right, with each CAB taking an input from the two cells immediately to its left, one from each row. The inputs and output of each CAB are also connected to pins of the chip package. In our case most of the external pins are left unconnected apart from to supply signals to the leftmost cells, to extract an output from the top rightmost cell and to provide some external resistors and capacitors, as shown.

Each CAB contains an operational amplifier surrounded by a few components with reconfigurable connections. If a cell's input is nominated from the same row A and from the other row B, then the repertoire of possible CAB behaviors is A+B, −B, A, log(A), exp(A), Half-Rectify(exp(A)), Aux(external components), and High-Impedance-OFF. For some of these there are scaling factors

---

[1] See "Totally Reconfigurable Analog Circuit—TRAC" Datasheet TRAC020LH, Issue 2, March 1999. Fast Analog Solutions Ltd, Zetex group, at *http://www.fas.co.uk*.



8.2 FIGURE The layout of the TRAC020 chip, with external components.

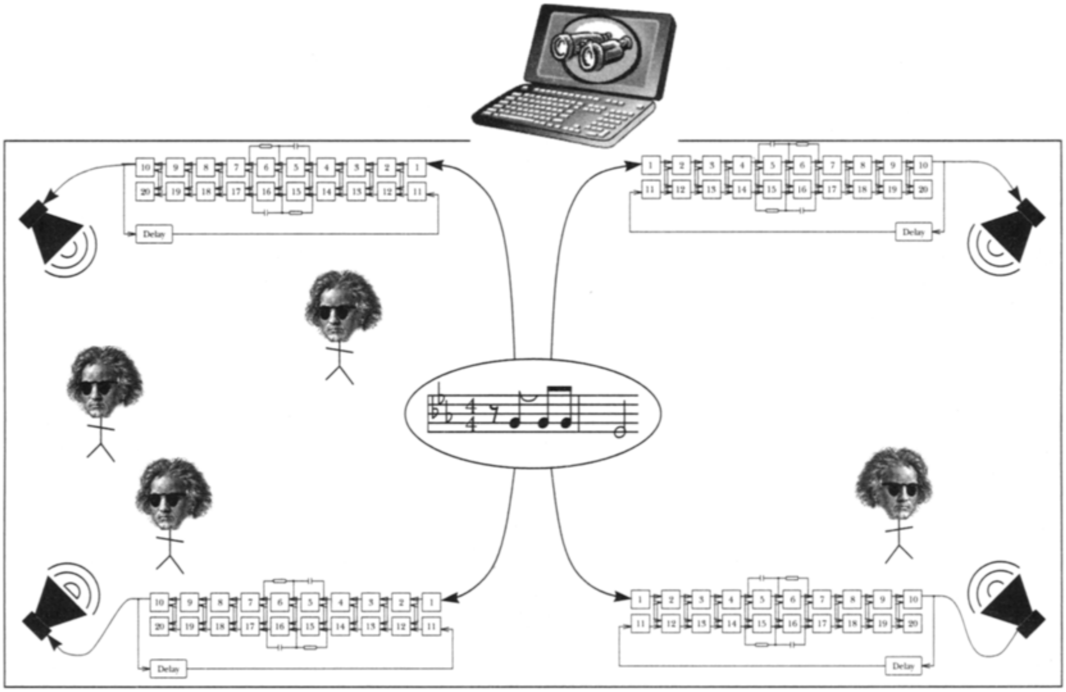
and offsets, given in the datasheet. The topology and CAB repertoire would in itself give very limited acoustic distortion effects, so some external resources were provided. For each TRAC chip, a separate digital delay module was connected as shown in the figure. The additional external capacitors and resistors shown provide the possibility of differentiation or integration of the signal if the pair of CABs they are connected across are configured in the combination (OFF, Aux).

8.3 GALLERY SETUP

Figure 8.3 is a sketch of the gallery arrangement. A central sound source is transmogrified by a different evolved circuit, embodied in the reconfigurable TRAC chips, on its way to each speaker. Participants' movements in the surrounding gallery space are influenced by their aural experience. A computer observes the ostensible appreciation of each speaker via some sensing system and assigns evolutionary fitnesses to the circuits accordingly. The computer maintains a subpopulation of circuits for each speaker, and schedules individuals for evaluation, whereupon they are sent as configurations to the chips.

8.3.1 Setting

The Sound Gallery, by virtue of its interactive and adaptive nature, would clearly work best in a site-specific context. The choice of site raises a number of aesthetic and technical issues that need to be clearly addressed. We shall here discuss a range of possible settings into which the Sound Gallery could be


10 9 8 7 6 5 4 3 2 1
20 19 18 17 16 15 14 13 12 11
Delay
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
Delay
10 9 8 7 6 5 4 3 2 1
20 19 18 17 16 15 14 13 12 11
Delay
1 2 3 4 5 6 7 8 9 10
11 12 13 14 15 16 17 18 19 20
Delay

8.3 FIGURE The gallery setup.

implemented. Some of the settings considered are actual locations around Brighton, where the Sound Gallery was developed. Other possible settings considered are more generalized.

The Beach

In many ways, the beach is an ideal setting for the Sound Gallery. Brighton beach is well endowed with public art and has been the site of several temporary sculpture exhibits and art events in the recent past. A well-positioned, visually stimulating sculpture would inspire curiosity, attracting a steady flow of individuals eager to explore and interact with it. We can imagine the Sound Gallery implemented as a ring of tall obelisk-type structures, with the four speakers mounted in the tops of each pillar. The strange appearance of this structure would attract attention and invite passersby to enter the ring and explore the soundscape within.

The Art Gallery

The Sound Gallery could also be implemented in a more traditional art gallery setting. Suitable galleries in Brighton include Fabrica, a converted church in the North Lanes that has recently hosted a number of site-specific art installation pieces. Speakers could be concealed high in the gallery's rafters, and the old church architecture would provide interesting acoustic effects and a serene ambience.

One problem is that the flow of people through art galleries can be rather inconsistent. Galleries are often empty or sparsely populated for extended periods of time. A gallery implementation of the Sound Gallery would therefore have to be tolerant to fluctuating attendance levels. This problem has been addressed by incorporating a fitness decay function into the algorithm, causing the fitness values of older genotypes to decay or age over time.

Architectural peculiarities of the gallery that encourage people to congregate in some part of the gallery more than in others could unfairly bias the system into rewarding a speaker positioned in that part of the gallery, regardless of the sounds it makes. A gallery implementation of the Sound Gallery would also have to take this into consideration.

The Nightclub

A club or festival setting could also provide an interesting environment for the Sound Gallery. The idea of wandering around a room and listening to semimusical sounds may make more sense to people in a club context than it does in an art gallery. This is essentially what people in nightclubs do anyway. Clubs also have the advantage of being occupied (usually) by a fairly large and fairly stable number of people, all of whom seem happy enough to mill about in the same room for hours on end. What better captive audience could one ask for?

8.3.2 Sensing Systems

The choice of sensing system needs to be tailored to the environment of the installation. In a nightclub setting, for example, enthusiasm levels may be betrayed by the vigor of the participants' bouncing in the vicinity of each speaker, which could be measured with pressure pads under the floor covering. In other environments the participants are likely to be less energetic but more free to move, so their distance from each speaker can be used. Since people may be unaware of their participation and should not necessarily have to be conscious of the preferences they express, asking them to score the speakers more directly is not an option.

There are several choices for sensors that can detect the distance or position of humans. Which is best depends on cost, time, the size and furnishing of the space, indoors or out, and any need to withstand attack. Cost-effective candidates are floor pads, passive infrared (sensing movements of body heat sources within an area, as is done for burglar alarms), sonar (ultrasonic time of flight), electric field proximity sensing (Russel Bik Design 1999), and image analysis from an overhead camera. At greater cost, sensors based on microwave Doppler shift or laser time of flight might be explored. Finally, in a well-controlled setting, if the participants are in on the game, they can each be given some sort of reflector, transponder, or beacon, allowing their position to be triangulated from base stations by radio or ultrasound.

In our first experiment, described later, one of the crude but cheap and robust sensing methods was emulated: a long-suffering human simply counted how many people were in each quadrant of the space and entered this directly into the computer.

See Borenstein et al. (1994) for a detailed discussion of the sensing systems considered above.

8.4 CONTEXTUALIZATION: ARTIFICIAL LIFE AND ART

This section places the Sound Gallery within the context of other existing works of A-Life art. First we shall discuss the work of artists who have used evolutionary algorithms in the production of visual art. We will then discuss some of the ways in which evolutionary algorithms have been applied to the creation of music. Following this, we will follow a brief discussion of interactive genetic art, and we shall also review some existing interactive and adaptive nongenetic artworks. For a detailed overview of existing artworks that incorporate ideas and techniques derived from the field of artificial life, and of some of the philosophical and aesthetic issues raised by these artworks, see Woolf (1999).

8.4.1 Evolutionary Algorithms and Visual Arts

Several artists have used evolutionary algorithms to create works of visual art. Perhaps the most well known are Karl Sims and William Latham. Both of these artists have used systems that allow the artist to control the evolution of new images through the allocation of fitness values based on subjective aesthetic criteria rather than by some hard-coded fitness function.

Sims used an interactive genetic algorithm that depicts each member of the current population, as an array of choices. The artist may then select from the possibilities embodied by the members of the current population, and genetic operators will act on the selected individuals to produce the next generation of images. As Sims explains, "Interactive evolution allows procedurally generated results to be explored by simply choosing those that are the most aesthetically desirable from each generation" (Sims 1991).

Color Plate 13 shows a generation of images from Karl Sims's system, ready for the artist to make his aesthetic selection governing the next stage in the evolutionary process. Color Plate 14 shows a completed image that was evolved with this system.

Sims has also produced computer animations and virtual creatures using evolutionary algorithm (EA) techniques. For more detailed information on Sims's work, see Sims (1991, 1994).

William Latham has used computer systems he developed with Stephen Todd to produce organic and geometrical forms. The programs he uses are called Mutator and Formsynth. These systems also allow the artist to control the evolution of images through aesthetic selection; however, the evolutionary algorithms used by Latham are implemented slightly differently from those used by Sims and control different image parameters. Latham's system allows the artist to evolve only selected aspects of the image being generated, for example, the degree of "twist" to be applied to some geometrical form. Sims's system, on the other hand, is more open ended and allows evolutionary processes a greater degree of control. For more detailed information on Latham's art and the computer systems he developed with Todd, see Todd and Latham (1992).

8.4.2 Evolutionary Algorithms and Music

The composer John Cage used chance operations to create many of his works. The piece "Music of Changes," composed in 1952, was written by "asking questions . . . detailed to all of the various aspects of a piece of music in as complicated and thorough a way as I [Cage] could" (Retallack 1996). Cage answered these questions by consulting the Chinese book of changes, the I Ching. In later works, Cage used the toss of a coin to make compositional decisions, and eventually he progressed onto the use of simple computer programs. Perhaps these experiments may be regarded as precursors to the use of evolutionary algorithms in musical composition. Allowing an EA to manipulate musical parameters such as note lengths, amplitude, frequency, duration, and spacing is not dissimilar to the use of chance operations as a means of answering complex musical questions.

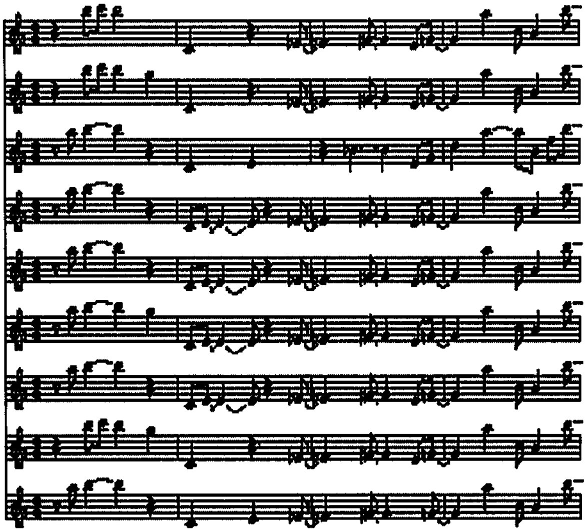
Gary Lee Nelson, of the Oberlin Conservatory of Music, has used a distributed genetic algorithm to produce musical compositions that he refers to as "Sonomorphs." Nelson's work is inspired by Dawkins's "Blind Watchmaker" program, and like the evolutionary art of Sims and Latham, uses the aesthetic judgments of the artist as fitness criteria. However, the use of aesthetic selection is far more problematic when applied to the evolution of music than it is when applied to the evolution of visual art. A visual interface like that used by Sims allows an entire generation to be evaluated simultaneously and within a negligible period of time. However, equivalent audio interfaces do not exist. As Nelson (1993) explains:

> Music presents a difficulty that we do not find in Dawkins's model. With The Blind Watchmaker program we can compare and evaluate an entire generation with a single glance. If all of the sonomorphs in a population were presented at once the cacophony would make it impossible to distinguish let alone choose the stronger individuals. Presenting sonomorphs sequentially taxes the memory even in (small) populations . . .

The temporal nature of music further hinders attempts to evolve compositions using aesthetic selection. If each individual must be listened to in full in order to be judged, evolutionary runs become lengthy and laborious processes that try the patience of even the most dedicated. Nelson's solution to these problems is to use a combination of audio and graphical feedback to aid the process of aesthetic selection. Figure 8.4 shows a graphical representation of a population of Sonomorphs evolved using Nelson's system.

Spector and Alpern have constructed a system they call GenBebop that uses genetic programming to compose jazz music. GenBebop attempts to overcome the problems associated with the artificial evolution of music mentioned above by using a neural network trained to act as a music critic to evaluate the individual programs. (For more information on their approach, see Spector and Alpern 1994, 1995.)

We have not attempted to bypass the problems identified by Nelson. Instead we exploit them. The genotypes of the Sound Gallery are not evaluated individually, but four at a time, for our aim is not to evolve a polished and well-composed piece, but to generate a pleasing and constantly evolving "cacophony" of sounds. It is true that a single listener may have difficulty distinguishing between individual voices, but the group dynamics of a larger number of participants, each one



8.4.
FIGURE

A population of Sonomorphs. From Nelson (1993), reproduced with permission.

simultaneously engaged in his or her own aural investigations, describes aesthetic selection enough to allow for evolution to occur. There is no single individual who evaluates each genotype in turn. Instead many different individuals evaluate several genotypes together. It is possible that, as a result of this, interesting symbiosis effects between the different genotypes, the sound source, and the group behavior of the participants may occur.

Others who have experimented with the use of evolutionary algorithms for creating musical compositions include John Mount at Carnegie Mellon University, Pittsburgh, and Jeffrey Putnam at the New Mexico Institute of Mining and Technology.

8.4.3 Interactive Genetic Art

All of the genetic artworks discussed above are interactive to some degree. The process of aesthetic selection requires human feedback, and evolved images (and sounds) can be seen as the products of an artist's interactions with the computer system. Given this concept, it is but a small step to the creation of evolutionary art systems that will accept feedback from a wider group of participants. The medium of the World Wide Web has made this possible. Several sites now exist that allow visitors to participate in a process of aesthetic selection. A population of evolving images is displayed by a visual interface much like that used by Sims, and visitors are given the opportunity to vote for their favorite images from the current generation. Those images that receive the most votes are then selected for breeding. One such Web-based interactive genetic art system is the "Evolutionary Art Bottle Breeder," created by the artist Jeffrey Ventrella. This is a Web site that allows visitors to participate in the breeding of images depicting bottles of Absolut Vodka.[2]

There have also been attempts to create interactive evolvable music composition systems. One of the more convincing of these is perhaps GenJam, created by John Biles. This system is supposedly able to interact with the music of human performers by translating "heard" sounds into genotypes that are used to evolve a musical "reply." (For more on GenJam, see Chapter 5 in this book and Biles 1998.)

8.4.4 Interactive, Adaptive, and Autonomous (Nongenetic) Artworks

In an introduction to the 1968 interactive artwork "The Colloquy of Mobiles," the cybernetician Gordon Pask wrote: "An aesthetically potent environment encourages the hearer or viewer to explore it, to learn about it, to form an hierarchy of concepts that refer to it; further it guides his exploration; in a sense, it makes him participate in, or at any rate see himself reflected in, the environment" (Pask 1968).

Pask cites "music that bears repetition" and "paintings worth seeing twice" as examples of passive environments endowed with this property of "aesthetic potency"; however, he clearly believes that the artistic use of active and reactive systems can enhance an environment's aesthetic potency.

This brings us to the concept of interactive and adaptive art. An interactive and adaptive artwork is an artwork that can respond and adapt to changes in its environment, having the potential to modify its behavior, for example, in response to the movements and sounds made by a human audience. A-Life artist Ken Rinaldo describes this as "a cybernetic ballet of experience, with the computer/machine and viewer/participant involved in a grand dance of one sensing and responding to the other" (Rinaldo 1998).

Interactive and autonomous artworks transform the conventional space of communication between artist, artwork, and audience and blur the traditional distinctions existing between these groups. Ihnatowicz's "Senster," created in 1969, was one of the first works to actively force this deconstruction of the traditional dichotomous divide between the static work of art and its passive, distanced audience. The Senster was a 16-foot articulating arm controlled by a Honeywell-8 computer. It was equipped with a variety of sensing instruments, including directional microphones, radar, and sonar. The Senster would come over to explore the sources of slight sounds or movements, and would back away from aggressive motions, responding to the changing activities of its audience in an eerily lifelike manner (see Figure 8.5). As the actions of individual spectators were variables that influenced the behavior of the Senster, a dialogue of participatory interplay between audience and artwork emerged whereby individuals culled meaning from the experience of interaction.

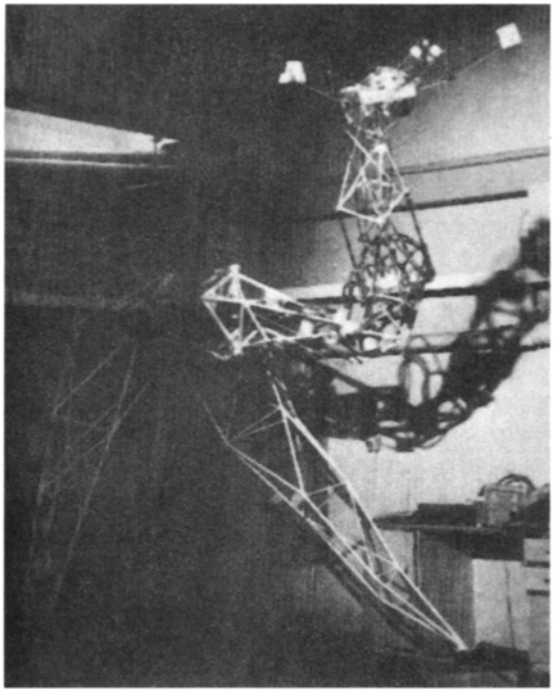
---

[2] *http://panushka.absolutvodka.com/kelly/8–0.html.*

Interplay between the personalities and behavior of human and machine, of audience and artwork, defines novel forms of artistic expression and experiences that are able to convey significance and meaning through activity and participation. Conventional artworks, on this model, are to be regarded as static and changeless unidirectional monologues whose meanings are always to be analyzed and interpreted from a safe critical distance. Autonomous, interactive, and adaptive artworks, on the other hand, engage the audience in an active dialogue, allowing meaning to be discovered through personal involvement. As Kac writes: "At its best, interactive art implies less stress on form (composition) and more emphasis on behaviour (choice, action), negotiation of meanings, and the foregrounding of the public who, now transformed into 'participants' acquire a prominent and active role in shaping their own experiences" (Kac 1994).

If interactive artworks transform spectators into participants, they also force us to question the conventional view of the artist as the author and communicator of meanings. Different participants may choose to interact with the same artwork in different ways, and the artwork may respond differently to distinct individuals accordingly. The mode of interaction chosen by particular individuals may therefore profoundly influence their aesthetic experience of the artwork, and so to some extent spectators themselves become the authors of their own aesthetic experience. Interactive artworks therefore actively encourage a kind of fragmentation and displacement of authorship, nurturing an aesthetic environment that fits well with the ideas of radical poststructuralist reader-response literary theorists like Stanley Fish, who assert that meaning is indeterminate and does not reside "in" the text but is constructed by the reader through active participation (Fish 1980).

Contemporary artists who create interactive, adaptive, and autonomous robotic artworks include Norman White, Simon Penny, and Ken Rinaldo. Norman White's robotic sculpture *The Helpless Robot* is an interactive artwork that, lacking the ability to move of its own accord, instead berates those detected in its vicinity to move it to a more "desirable" location (see Color Plate 9). Simon Penny has



8.5 FIGURE Ihnatowicz's interactive artwork "The Senster" (Reichardt 1978).

created an autonomous robotic artwork called *Petit Mal*.[3] Ken Rinaldo has built a group of musical interactive sound sculptures called *The Flock* with the ability to communicate to each other using a language of telephone dialing tones and that exhibit simple flocking behavior.[4]

Ken Sharman et al. (1998) are working on an artistic system that is interactive in a rather more direct sense; their aim is to develop a system that will convert the participant's thoughts into music, using artificial neural networks to "extract musical thought features from the human EEG."

8.5 THE SOUND GALLERY ALGORITHMS

In this section the software and genetic algorithms that we have developed for the Sound Gallery project will be described.

---

[3] *http://www-art.cfa.cmu.edu./www-penny/.*

[4] *http://ylem.org/artists/krinaldo/emergent1.html.*

8.5.1 Two-Phase Hill-Climbing/ Island Model GA

The algorithm actually works in two distinct phases, as shall be described in more detail below. First a hill-climbing phase is implemented, and then a steady-state island model genetic algorithm. The hill-climbing algorithm is used to initialize the four islands (subpopulations) associated with the chips. Hill-climbing may start from random initial circuits, from hand-designed circuits, or from randomly generated circuits that have been selected by the user. Hill-climbing creates initial populations for the genetic algorithm comprised of reasonably fit diverse solutions that are already related to each other to some degree.

The second phase of the algorithm begins with the implementation of an island model genetic algorithm. An island model GA, as defined by Whitley et al. (1997), is a GA that uses several processors to explore the same problem space simultaneously and in parallel, while periodically exchanging genotypes between machines to enable the sharing of information. Our GA only uses one processor, but does maintain four independent subpopulations between which individuals migrate at times. The island model GA was chosen so that each speaker would be driven through circuits evolved from different subpopulations, each having their own individual genetic characteristics, hence reducing the likelihood of all the speakers producing similar sounds. It was also hoped that the migration operator used with island model GAs would cause interesting effects to occur; allowing, for example, aesthetically interesting sounds to "jump" from one speaker to the next.

8.5.2 Hill-Climbing Phase

Four initialization genotypes are generated, one for each of the four subpopulations. Hill-climbing then commences, with each subpopulation working in parallel to the other three. The initialization genotypes undergo repeated mutations, generating new genotypes that represent new TRAC configurations. Each new genotype is evaluated and assigned a fitness value. When a mutation is evaluated to be fitter than, or equally fit to, the parent genotype from which it was derived, then this mutant genotype is stored as the next member of its subpopulation and will be used as the source for subsequent mutations.

The hill-climbing algorithm is run until the four island populations are filled to capacity. This provides the genetic algorithm with its initial populations, each individual of which is already associated with a fitness value. These fitness values could not have been calculated in the conventional manner, as they represent a conglomeration of the subjective opinions and random wanderings of those present in the gallery, and cannot be mathematically derived from any aspect of individual genotypes or phenotypes. From this initial condition, the steps of the island model GA are followed and repeated (forever).

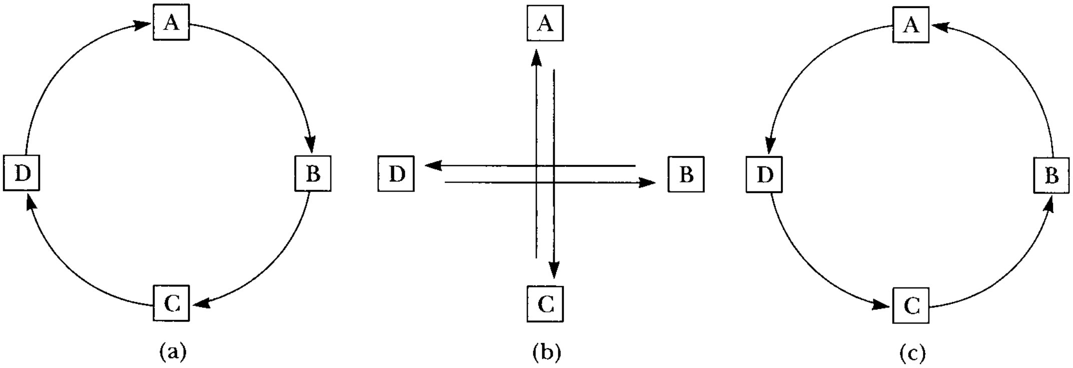
8.5.3 Island Model Genetic Algorithm Phase

Linear rank-based selection is used to select two "parent" genotypes from each island subpopulation. Child genotypes are derived from each pair of parents through the application of crossover, mutation, and replication genetic operators. The TRAC development board is then reconfigured so that the circuit specifications represented by each of the new child genotypes are physically manifested in silicon. Each of the four circuits on the TRAC development board are then allocated fitness values, and the new genotypes replace the least fit members of their respective island subpopulations. This sequence of events represents one iteration of the algorithm.

The Sound Gallery GA also implements a migration scheme similar to that developed by Whitley et al. Every *X* iterations, each island copies an elite group of the fittest individuals from within its own population to one of the other islands. The receiving island avoids overpopulation by deleting the same number of the least fit individuals from within its own ranks. Islands are paired differently for each iteration of this migration cycle, so that they send and receive individuals from different sources throughout the duration of the evolutionary run. Figure 8.6 illustrates different orders of migration strategy that we have used.

8.5.4 The Need for Aging

Early experiments with small island populations showed that the design of the genetic algorithm allowed very fit genotypes to remain in the population indefinitely. Less fit, more recent additions to the population could be eradicated by the periodic migrations of fitter strings from neighboring islands. This situation could lead to population convergences—an unwelcome prospect because, apart from the obvious desire to maintain genetic diversity in the populations, it also seemed likely that some of these fitter genotypes could be the products of freak evaluation conditions. There could be times, for example, when a particular genotype receives a very good fitness value simply because of some freak fluctuation in audience movements, and not because it represents a circuit of



8.6 FIGURE Migration strategies: (a) first order, (b) second order, and (c) third order.

particular merit. This could easily occur, for instance, at times when there are only one or two persons present in the gallery setting. Because genotypes are only evaluated once, and because this evaluation may lose validity with increasing temporal distance from its context, it was felt necessary to allow the genetic algorithm some means of recovery from freak convergence conditions. A periodic "aging" of fitness values was therefore introduced. The fitness values of each member of all island populations are periodically incremented by a small amount, making them more costly. This has the effect of making those genotypes that have been in the population for a large number of generations increasingly less fit, so that freakishly "good" genotypes cannot linger indefinitely in the highest echelons of the population. They are instead quietly retired, to make room for fresh young strings.

8.5.5 Encoding Scheme

A very simple encoding scheme has been used. Each genotype is a string of integer values, each sequentially corresponding to a single CAB of a TRAC chip. Each node of the genotype is allocated a value corresponding to the function type to which its associated CAB is to be configured. This encoding scheme enables all possible TRAC configurations to be represented as a string of 20 integers whose values may vary between 0 and 7.

8.5.6 The Fitness Function

Conventional genetic algorithms are often used to find an optimal solution for some well-defined problem domain. A traditional favorite GA domain is "The Traveling Salesman Problem" (TSP), where the goal is to find the most efficient path of travel between a number of unevenly distributed nodes, where each node is visited once and once only. In such optimization problems, finding the fitness of a solution normally involves evaluating the optimality of some property of its phenotype. For the TSP, for example, the fitness of a solution is normally calculated as the total length of the path encoded by the evolved genotype.

The GA utilized for this project perhaps has more in common with natural selection than it does with algorithms used for optimization problems. Ultimately, whether or not a particular genotype is selected for depends not on some calculated value, but on how suited that genotype is to its current environment. This will depend on the process of interactions of all in the space concerned, and individual motivations may vary. Fitness of a genotype will depend on the positions of human participants in the gallery. However these participants may not be making conscious decisions and choices, like those made by the Web art voters discussed above, but will register their votes in a more subliminal level. People in the gallery space need not even be aware of the fact that their each and every movement has a direct bearing on the evolution of new circuits, as omniscient sensors track their changes in position over time . . .

The fact that, at each new step of the algorithm, four new child genotypes will be simultaneously evaluated means that the GA effectively places these new genotypes in direct competition. Each new genotype is evaluated in the context provided by the others. If the same genotype reemerges in a different context, it is possible that it may be interpreted differently by the human participants, and hence receive a different fitness score. The fact that the geographical location of participants controls fitness evaluation also makes possible a kind of implicit fitness sharing, as a participant positioned equidistantly between two rival speakers will reward the genotype of the circuits driving each of these speakers equally.

Fitness evaluations may also be affected by noise in the measurements provided by devices tracking positions of human participants, by the time lengths allowed for each evaluation, and by idiosyncrasies in the sound source.

8.5.7 galSim

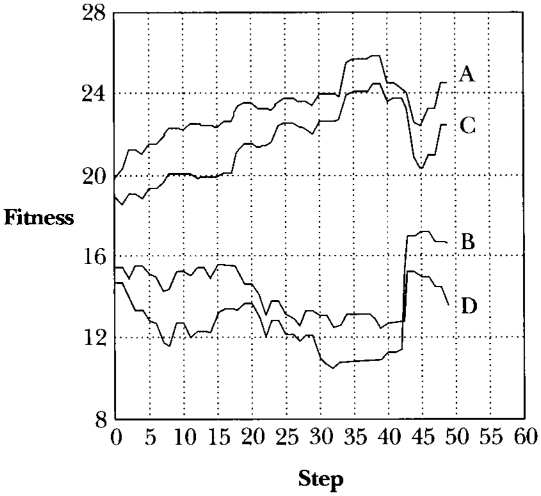
For the purposes of development and testing, a program called galSim that simulates the movements of a number of people as they navigate an artificial gallery space was written. This program is used to provide the genetic algorithm with automatically generated fitness values (Figure 8.7), making it possible to run experiments with the system outside of the gallery setting. This makes unnecessary the difficult task of recruiting patient volunteers to "interact" with the system while the code is fine-tuned.

A lattice gas cellular automaton was used to generate the model. The gallery is represented by a two-dimensional grid, the dimensions of which can be altered by the user to best simulate different gallery spaces (Figure 8.8).

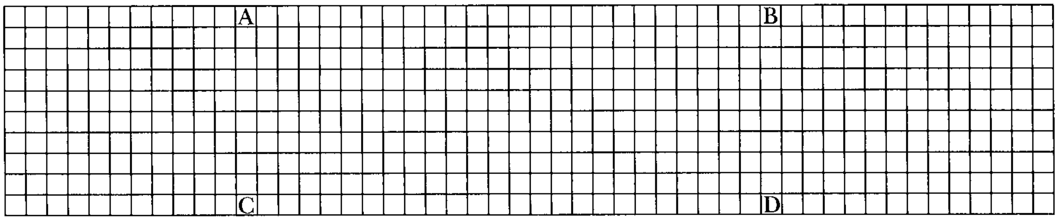
Each individual is represented as a point within the gallery grid. Each individual is also associated with a number of variables that determine his or her current status. The values of these variables represent the position of the individual within the gallery and also indicate which of the four speakers is that individual's current favorite. With the passing of simulated time, these values undergo mutations designed to approximate the movements and reactions of real people within a real gallery space.

8.6 THE EXPERIMENT

An experimental run of the prototype Sound Gallery system developed for this project was executed in a small room in the Centre for Computational Neuroscience and Robotics (CCNR). The four speakers were positioned in each corner of the room, each one connected to one of the TRAC020 chips on the development board. Each chip received an identical input signal, a repetitive audio loop comprised of low frequency drum beats, high frequency cymbals, and synthesized notes of various frequencies, produced with drum-machine simulation software running on a laptop computer. The Sound Gallery program was running on a second laptop computer, and this computer was connected to the development board by means of a parallel cable. The room was divided into quadrants, with the dividing lines marked clearly on the floor of the room (Figure 8.9). Each speaker was allocated one quadrant. Volunteers were instructed to explore the sounds generated by each speaker, and to settle in the quadrant allocated to whichever speaker they felt produced the most aesthetically pleasing sounds; this process was to be repeated each time the circuits were reconfigured.



8.7 FIGURE Fluctuating fitness values returned by the galSim program over 50 steps.



8.8 FIGURE The cellular automaton gallery simulation grid, complete with speaker locations, as generated by the default galSim settings.

Before the start of the experiment, each chip was configured to a randomly generated circuit selected for its aesthetic value. The hill-climbing algorithm then used these circuits as its starting point, and hill-climbing continued for 10 steps. After 10 steps of hill-climbing, the island model genetic algorithm was implemented and continued running for a further 40 steps.

Fitness values for each evolved circuit were entered into the computer running the Sound Gallery program by hand. To make this task executable, the fitness of a circuit was defined as $(T - Q)$, where T was the total number of volunteers participating in the experiment, and Q was the number of volunteers who settled in the quadrant allocated to the speaker driven by the circuit under

8.9 FIGURE The grid marking positions of participants.

consideration. The experiment ran for about an hour and a half. During this time, the number of volunteers present fluctuated between five and eight.

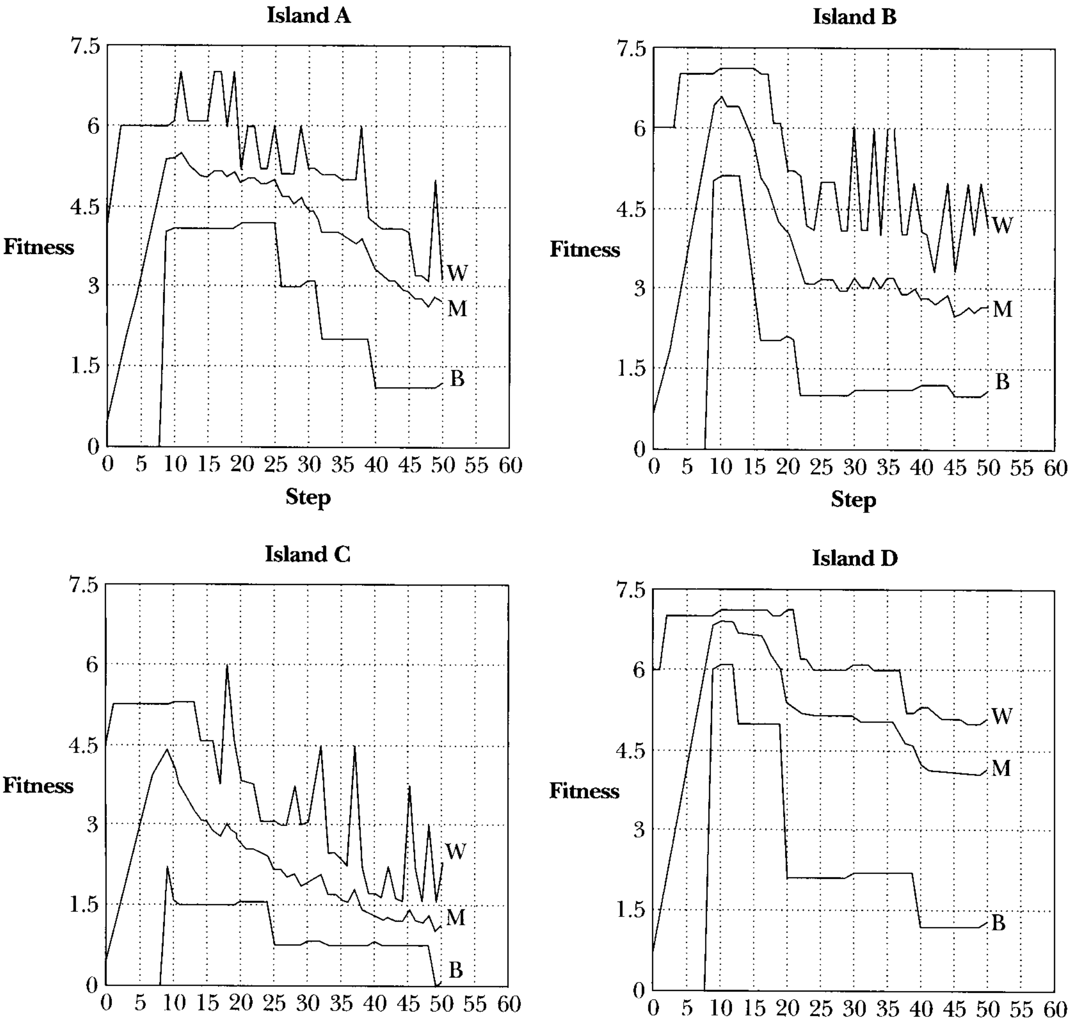
8.6.1 Results

The volunteers responded well to the concept of exploring the different sounds emerging from different parts of the room and seemed to enjoy and be genuinely enthusiastic about the experience. The Sound Gallery proved itself capable of producing a fairly large repertoire of interesting distortion effects, and when each chip distorted the sound to each speaker differently, a pleasing cacophony of related but highly individualized reverberations filled the room. To describe the resultant aural chaos as "music" may be to somewhat labor the point—indeed, one of those present thought it more redolent of the sounds that brainwashed Harry Palmer in *The Ipcress Files* than of anything more euphonious. However, with the four TRAC chips perpetually improvising new adaptations to their most popular configurations, and with the four speakers continually emitting their distorted variations to the sound source's leading theme, it seems fair to claim that the "repetition and variation" required to fulfil Schoenberg's definition was achieved.

Some interesting artifacts of the way the system was implemented became apparent during our experiment. About halfway through, speaker D stopped producing sounds, and speaker C became very quiet. This revealed an interesting idiosyncrasy: participants were actually drawn to the quieter speaker, and crowded around it in an attempt to hear the sounds it produced. This effectively gave the quiet speaker an abnormally good fitness value, and so in the next stage of evolution, that speaker became quieter still, until in fact it became barely audible.

Since only Speakers A and B were producing interesting sounds, for the rest of the run, people generally boycotted the quadrants allocated to speakers C and D, only moving between the quadrants allocated to A and B. Thus every individual generated by the genetic algorithm from the populations of island C and island D effectively received the same fitness score. This meant that proper selection could not take place, and the sounds produced by speakers C and D did not improve. The only time C and D could have benefited from evolution would have been when receiving high fitness migrants from other islands. In practice, such migrations were too few and far between to have much beneficial effect. Our desire to retain the genetic diversity of each island had driven us to set the migration rate to just one individual every 20 steps. This, it would seem, was rather too low. As can be seen from Figure 8.10, the fitness of the best individual in island D does improve dramatically every 20 steps, that is, when a fit genotype is received from one of the other islands. This in practice had little effect; any benefit introduced by the new fit genotype was immediately eradicated, probably as a result of crossover with lesser members of the population. Figure 8.10 clearly illustrates island A and B's roles as the source of most of the innovative and interesting sounds. The best individuals on these islands improve in sudden dramatic steps, and not at step numbers that are multiples of twenty, the points at which migrations occurred. The spikiness of the worst individual curves reflects those times when evolution produced a particularly repellent sound, causing the participants to boycott that island. Note that the worst individual line for island D is relatively flat, reflecting the fact that this island was fairly consistently boycotted.

The steep rise (worsening) in mean fitness value between steps 0 and 10 for all islands is due to the fact that all islands are initialized with a population of 10 null strings with fitness values of zero. With each new step of hill-climbing, these null strings are gradually replaced with real genotypes that have real fitness values. The best fitness value for each island also peaks at 10 steps for this same reason, as before this point the best value will always be zero. After this point, all the



8.10 FIGURE Worst, mean, and best fitness curves for each of the four islands.

fitness values for each island gradually decline (improve) as evolution replaces weaker members of the populations.

8.7 CONCLUSIONS

The goal of this project has been to develop a working prototype of the Sound Gallery system that would be capable of producing sounds interesting and varied enough to capture the attention and imaginations of a group of volunteer participants. We feel satisfied that this goal has been achieved. The experiment described above stimulated both enthusiasm and scientific curiosity and kept a roomful of people entertained for an hour and a half. During this time the dingy backroom of the CCNR was transformed into a perpetually mutating soundscape that at every moment offered up new aspects of itself for exploration. The Zetex TRAC020 chip, working in concert with the digital delay units, proved itself capable of producing a reasonably large and interesting repertoire of distortion effects. It was particularly good at picking out very high and very low frequencies from the source signal, and of distorting these extreme frequency bands in interesting ways. A major weakness was the fact that if a speaker repeatedly failed to produce interesting sounds, became very quiet, or produced no sound at all, and hence was continually boycotted by the participants, aspects of the design of the GA made it very difficult for such a speaker to recover. Steps may be taken in future implementations of the Sound Gallery to exploit or eradicate the effects of these idiosyncracies.

We intend to continue working on this project and eventually to get the Sound Gallery up and running as an interactive installation art exhibit in Brighton and elsewhere. Further work will include the development of some form of sensory apparatus capable of automatically generating the data needed for fitness evaluations. Extra reconfigurable switches will also be introduced to allow for external components to be attached to the TRAC development board, and for the use (or nonuse) of these external components to be placed under evolutionary control. Parts of the circuitry from the digital delay units may also be cannibalized and then reconnected to the main circuit via reconfigurable switches, allowing parameters such as the type of delay effect and the length of delay effect used, which had to be fixed for the purposes of this prototype project, to be placed under evolutionary control. Some aspects of the genetic algorithm controlling evolution may also be modified. Collaborations with Brighton-based architects, sculptors, and musicians have also been discussed, and the artistic input of these talented people will enhance the potential of the Sound Gallery and help to place it in a functional and aesthetically stimulating environment.

ACKNOWLEDGMENTS

Many thanks to Zetex, Jon Stocker, Ian Oszvald, and the CCNR.

REFERENCES

Biles, J. (1998). Interactive GenJam: Integrating Real-Time Performance with a Genetic Algorithm. In *Proceedings of the 1998 International Computer Music Conference,* ICMA, San Francisco.

Borenstein, J., et al. (1994). *Where Am I? Sensors and Methods for Autonomous Mobile Robot Positioning.* University of Michigan. Technical Report UM-MEAM-94-21.

De Garis, H. (1995). *Evolvable Hardware Workshop Report,* ATR, Kyoto. *http://www.cwi.nl/~ehrenbur/EvolvableHardware/WorkshopReport.html.*

Dollhausen, J. (2000). Interactive Art Machines. *http://www.wsu.edu/~jackdoll/jak/qanda/sense/sense.htm.*

Fish, S. (1980). *Is There a Text in This Class?* Harvard University Press.

Flockton, S., and K. Sheehan (1998). Intrinsic Circuit Evolution Using Programmable Analogue Arrays. In *Pro. 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES '98). Springer-Verlag.

Flockton, S., and K. Sheehan (1999). A System for Intrinsic Evolution of Linear and Non-Linear Filters. In *Proceedings of The First NASA/DoD Workshop on Evolvable Hardware,* Pasadena, California.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimisation and Machine Learning.* Addison-Wesley.

Higuchi, T., M. Iwata, and L. Weixin (eds.) (1997). *Proc. 1st Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES'96). LNCS 1259, Springer-Verlag.

Holland, J. (1975). *Adaption in Natural and Artificial Systems.* University of Michigan Press.

Horowitz, P., and W. Hill (1980). *The Art of Electronics.* Cambridge University Press.

Kac, E. (1994). Telepresence Art. In R. Kriesche (ed.), *Teleskulptur,* Kulturdata, pp. 48–72. *http://www.ekac.org/Telepresence.art._94.html.*

Layzell, P. (1998). The "Evolvable Motherboard"—A Test Platform for the Research of Intrinsic Hardware Evolution. CSRP 479, January, School of Cognitive and Computing Sciences, University of Sussex.

Miller, J. (1999). On the Filtering Properties of Evolved Gate Arrays. In *Proceedings of The First NASA/DoD Workshop on Evolvable Hardware,* Pasadena, California.

Miller, J., A. Thompson, T. Fogarty, and P. Thomson (eds.) (2000). *Proc. 3rd Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES2000). Springer-Verlag.

Motorola (1997). Motorola Semiconductor Technical Summary. Introducing Motorola's Field Programmable Analogue Array. *http://www.mot.com/SPS/.*

Nelson, G. (1993). Sonomorphs: An Application of Genetic Algorithms to the Growth and Development of Musical Organisms. *http://timara.con.oberlin.edu/people/~gnelson/papers/morph93/morph93.htm.*

Oldfield, J. V., and R. C. Dorf (1995). *Field Programmable Gate Arrays: Reconfigurable Logic for Rapid Prototyping and Implementation of Digital Systems.* Wiley.

Ozsvald, I. (1998). *Short-Circuiting the Design Process: Evolutionary Algorithms for Circuit Design Using Reconfigurable Analogue Hardware.* KBS MSc thesis, School of Cognitive and Computing Sciences, University of Sussex.

Pask, G. (1968). Machines and Environments. The Colloquy of Mobiles. In J. Reichardt (ed.), *Cybernetic Serendipity—The Computer and the Arts,* Studio International.

Reichardt, J. (1978). *Robots: Fact, Fiction and Prediction.* Thames & Hudson.

Retallack, J. (ed.) (1996). *'Musicage'—John Cage in Conversation with Joan Retallack.* Wesleyan University Press.

Rinaldo, K. (1998). Technology Recapitulates Phylogeny. *Leonardo* 6(2). *http://mitpress.mit.edu/e-journals/LEA/ARTICLES/alife1.html.*

Rozmiarek, A. (1995). Global Positioning System. *Wired* 3 (October): 10.

Russel Bik Design (1999). *Electric Field Proximity Sensor Manual* 9-5-99. *http://www.bik.com.*

Schwefel, H., and G. Rudolph (1995). Contemporary Evolution Strategies. In F. Moran et al. (eds.), *Advances in Artificial Life* (Lecture Notes in Artificial Intelligence, vol. 929), Springer-Verlag.

Sharman, K., et al. (1998). *Music from Brainwaves.* Working Report, Centre for Music Technology, University of Glasgow. *http://www.elec.gla.ac.uk/~kenshar/.*

Sims, K. (1991). Artificial Evolution for Computer Graphics. In *Computer Graphics (Siggraph '91 Proceedings)* 25(4), July.

Sims, K. (1994). Evolving 3D Morphology and Behaviour by Competition. In *Artificial Life IV Proceedings.* MIT Press.

Sipper, M., D. Mange, and A. Perez-Uribe (eds.) (1998). *Proc. 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES'98). LNCS 1478. Springer-Verlag.

Spector, L., and A. Alpern (1994). Criticism, Culture, and the Automatic Generation of Artworks. In *Proceedings of the Twelfth National Conference on Artificial Intelligence,* AAAI-94, AAAI Press/MIT Press.

Spector, L., and A. Alpern (1995). Induction and Recapitulation of Deep Musical Structure. In *Working Notes of the IJCAI-95 Workshop on Artificial Intelligence and Music.*

Thompson, A. (1996) Silicon Evolution. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), *Genetic Programming 1996: Proc. 1st Annual Conf.* (GP96), MIT Press, pp. 444–452.

Thompson, A. (1997). Artificial Evolution in the Physical World. In T. Gomi (ed.), *Evolutionary Robotics: From Intelligent Robots to Artificial Life* (ER'97), AAI Books, pp. 101–125.

Thompson, A., and P. Layzell (1999). Analysis of Unconventional Evolved Electronics. *Communications of the ACM* 42(4):71–79.

Thompson, A., and P. Layzell (2000). Evolution of Robustness in an Electronics Design. In J. Miller, A. Thompson, T. Fogarty, and P. Thomson (eds.), *Proc. 3rd Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES2000), Springer-Verlag.

Thompson, A., P. Layzell, and R. S. Zebulum (1999). Explorations in Design Space: Unconventional Electronics Design through Artificial Evolution. *IEEE Trans. Evol. Comp.* 3(3):167–196.

Todd, S., and W. Latham (1992). *Evolutionary Art and Computers.* Academic Press.

Whitley, D., et al. (1997). Island Model Genetic Algorithms and Linearly Separable Problems. In *Proceedings of the AISB Workshop on Evolutionary Computation 1997.*

Woolf, S. (1999). Artificial Life and Art. In *Proceedings of the 3rd World Multiconference on Systemics, Cybernetics and Informatics and the 5th International Conference on Information Systems Analysis and Synthesis,* Orlando, Florida, vol. 8.

Xilinx. Inc (1996). XC6200 Advanced Product Specification V1.0. June 1996. In *The Programmable Logic Data Book. http://www.xilinix.com.*

Zebulum, R., P. Pacheco, and M. Vellasco (1996). Evolvable Hardware Systems: Taxonomy, Survey and Applications. In *Proc. 2nd Int. Conf. on Evolvable Systems; From Biology to Hardware* (ICES '96), Springer-Verlag, pp. 344–358.

Zetex TRAC020 (1999). Totally Reconfigurable Analogue Circuit. TRAC Issue 2, March, Fast Analogue Solutions, Zetex Group. *http://www.fas.co.uk.*

III
PART

CREATIVE EVOLUTIONARY DESIGN

9 **Creative Design and the Generative Evolutionary Paradigm**
John Frazer

10 **Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results**
John R. Koza, Forrest H. Bennett III, David Andre, Martin A. Keane

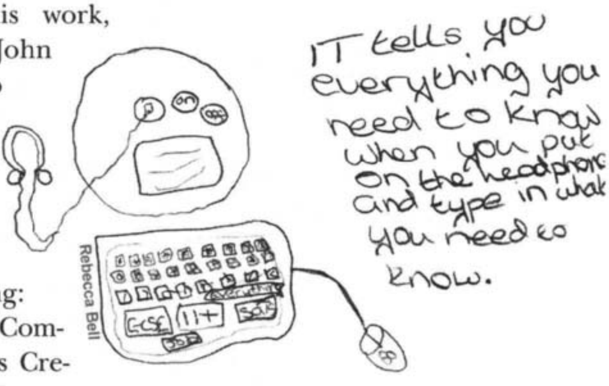
11 **Toward a Symbiotic Coevolutionary Approach to Architecture**
Helen Jackson

12 **Using Evolutionary Algorithms to Aid Designers of Architectural Structures**
Peter von Buelow

This part provides one of the links to the sister book *Evolutionary Design by Computers* (Morgan Kaufmann, 1999) by returning to the subject of creative evolutionary design. As before, this part explores the abilities of evolution to aid designers by generating novel and "creative" design solutions. Here we see both types of creative evolutionary system: methods designed to expand the creativity of designers and methods that produce novel and creative solutions automatically.

Chapter 9, "Creative Design and the Generative Evolutionary Paradigm," is provided by one of the earliest and most renowned pioneers in this field, John Frazer of the Hong Kong Polytechnic University. John outlines some of his work, from his infamous "John Conway days" up to the present day, describing his views of generative evolution for design problems.

The next chapter, "Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results," is provided by John Koza of Stanford University and his team. In this chapter, John continues the description of his work provided in the sister book, by exploring the subject of human-competitive results in the context of using GP to evolve new analog circuit designs.



IT tells you everything you need to know when you put on the headphone and type in what you need to know.

Rebecca Bell

Helen Jackson writes Chapter 11, "Toward a Symbiotic Coevolutionary Approach to Architecture," in which she investigates the use of Lindenmeyer systems with GP for evolving useful and novel architectural forms.

Finally, Chapter 12, "Using Evolutionary Algorithms to Aid Designers of Architectural Structures," by Peter von Buelow of the University of Stuttgart, shows how evolution can be used within an Intelligent Genetic Design Tool, enabling designers to explore new and creative solutions to design problems.

9
CHAPTER

Creative Design and the Generative Evolutionary Paradigm

John Frazer Hong Kong Polytechnic University

9.1 INTRODUCTION

Design: The imaginative jump from present facts to future possibilities.

Design as seen from the designer's perspective is a series of amazing imaginative jumps or creative leaps. But design as seen by the design historian is a smooth progression or evolution of ideas that seems inevitable with hindsight. It is a characteristic of great ideas that they seem self-evident and inevitable after the event. But the next step is anything but obvious for the artist/creator/inventor/designer stuck at that point just before the creative leap. They know where they have come from and have a general sense of where they are going, but often do not have a precise target or goal. This is why it is misleading to talk of design as a problem-solving activity—it is better defined as a problem-finding activity. This has been very frustrating for those trying to assist the design process with computer-based, problem-solving techniques. By the time the problem has been defined, it has been solved. Indeed the solution is often the very definition of the problem.

Design must be creative—or it is mere imitation. But since this crucial creative leap seems inevitable after the event, the question must arise, can we find some way of searching the space ahead? Of course there are serious problems of knowing what we are looking for and the vastness of the search space. It may be better to discard altogether the term "searching" in the context of the design process:

> Conceptual analogies such as *search, search spaces* and *fitness landscapes* aim to elucidate the design process. However the vastness of the multidimensional spaces involved makes these analogies misguided and they thereby actually

> result in further confounding the issue. The term *search* becomes a misnomer since it has connotations that imply that it is possible to find what you are looking for. In such vast spaces the term *search* must be discarded. Thus, any attempt at *searching* for the highest peak in the *fitness landscape* as an optimal solution is also meaningless. Furthermore, even the very existence of a *fitness landscape* is fallacious. Although alternatives in the same region of the vast space can be compared to one another, distant alternatives will stem from radically different roots and will therefore not be comparable in any straightforward manner (Janssen 2000).

Nevertheless we still have this tantalizing possibility that if a creative idea seems inevitable after the event, then somehow might the process be reversed? This may be as improbable as attempting to reverse time. A more helpful analogy is from nature, where it is generally assumed that the process of evolution is not long-term goal directed or teleological. Dennett points out a common misunderstanding of Darwinism: the idea that evolution by natural selection is a procedure for producing human beings. Evolution can have produced humankind by an algorithmic process, without its being true that evolution is an algorithm for producing us. If we were to wind the tape of life back and run this algorithm again, the likelihood of "us" being created again is infinitesimally small (Gould 1989; Dennett 1995). But nevertheless Mother Nature has proved a remarkably successful, resourceful, and imaginative inventor generating a constant flow of incredible new design ideas to fire our imagination. Hence the current interest in the potential of the evolutionary paradigm in design.

These evolutionary methods are frequently based on techniques such as the application of evolutionary algorithms that are usually thought of as search algorithms. It is necessary to abandon such connections with searching and see the evolutionary algorithm as a direct analogy with the evolutionary processes of nature. The process of natural selection can generate a wealth of alternative experiments, and the better ones survive. There is no one solution, there is no optimal solution, but there is continuous experiment. Nature is profligate with her prototyping and ruthless in her elimination of less successful experiments. Most importantly, nature has all the time in the world.

As designers we cannot afford profligate prototyping and ruthless experiment, nor can we operate on the time scale of the natural design process. Instead we can use the computer to compress space and time and to perform virtual prototyping and evaluation before committing ourselves to actual prototypes. This is the hypothesis underlying the evolutionary paradigm in design (1992, 1995).

9.2 THE ADAPTIVE MODEL FROM NATURE

A handful of scientists have defined theoretical frameworks where adaptation is the progressive modification of a given structure by the repeated action of certain operators. In nature, chromosomes are commonly the structure, and mutation and recombination are the operators.

This forms the basis of a mathematical formalism that defines a set of structures appropriate to the field of interest using all possible combinations of the elements (chromosomes). An environment is then defined for the system undergoing adaptation: This is an adaptive plan that determines successive structural modifications in response to the environment and measures the performance of different structures in the environment. The adaptive plan produces a sequence of structures from a set of operators. Information obtained from the environment influences the selections made. Evaluation then takes place using the specified measures.

Holland illustrates applications in genetics, economics, game playing, pattern recognition, control and function optimization, and the central nervous system. This treatment is then developed into generalized reproductive plans and genetic operators. This approach has come to be described as the application of "genetic algorithms" (Holland 1975; Frazer 1995).

9.3 THE GENERATIVE EVOLUTIONARY PARADIGM

Thus we define the generative evolutionary paradigm for creative design as follows: First, the problem needs some form of generic representation. Second, this representation is described in a genetic code. Third, a population of code scripts is created. Fourth, designs are generated from the genetic code scripts through some form of epigenetic development in an environment. Fifth, designs are evaluated, and the most successful are selected. Sixth, the selected code scripts are transformed by genetic operators such as crossover and mutation. The process is then repeated from step four. At some point the process may be stopped (see Figure 9.1).

Most engineering applications of the evolutionary approach are interested in convergence on an optimal solution in response to clearly defined computable criteria for selection. For the purposes of this chapter, we define this as *convergent evolution by natural selection* (noting that the real meaning of "convergent

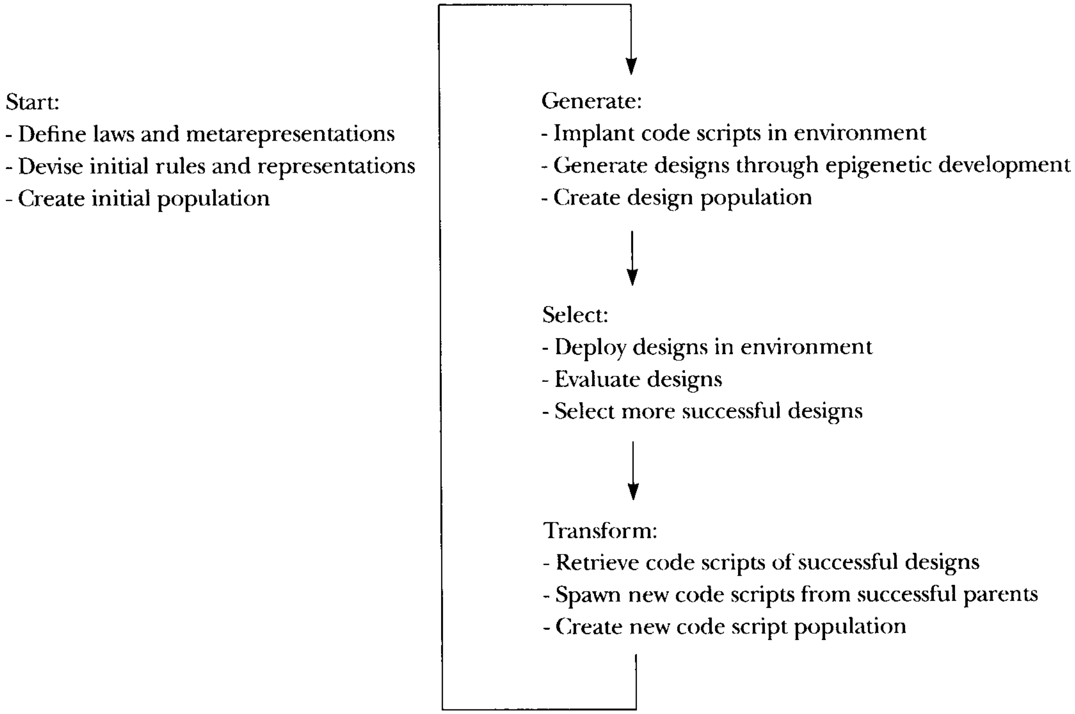

Start:
- Define laws and metarepresentations
- Devise initial rules and representations
- Create initial population
Generate:
- Implant code scripts in environment
- Generate designs through epigenetic development
- Create design population
Select:
- Deploy designs in environment
- Evaluate designs
- Select more successful designs
Transform:
- Retrieve code scripts of successful designs
- Spawn new code scripts from successful parents
- Create new code script population

FIGURE 9.1 The generative evolutionary paradigm.

evolution," as defined in biology, is somewhat different). But there are other possibilities.

In *On the Origin of Species*, Darwin talks first of the technique of artificial selection as practiced by the racehorse or dog breeder. In our model, *artificial selection* by the designer or user opens up the opportunity to demonstrate preference. It can be useful as a means of dealing with ill-defined selection criteria, particularly user-centered concerns. It also provides the opportunity for designers to use their experience and intuition to jump to faster results. Nature also exploits divergence to keep a varied gene pool active to cope with sudden changes such as the increasing success of a predator or change in the environment. So too, our model provides for *divergent evolution* for the generation of alternative ideas. This gives a matrix of four possible combinations of *natural/artificial selection* and *convergent/divergent evolution*, as in Figure 9.2 (Dawkins 1986; Sims 1991; Graham, Frazer, and Hull 1993, 1995).

	Convergent evolution	Divergent evolution
Natural selection	●	●
Artificial selection	●	●

9.2
FIGURE
Matrix of four possible evolutionary combinations.

9.4 PROBLEMS WITH THE PARADIGM

The process described above has had some demonstrated successes and holds great promise for future developments. It is also fraught with problems.

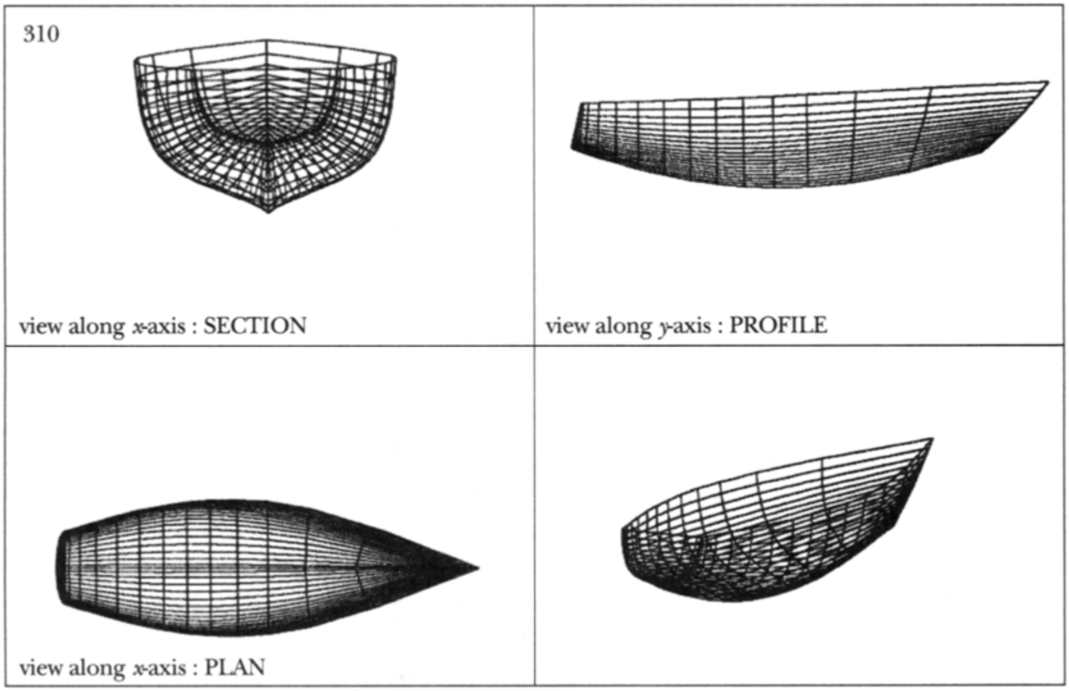
In step one, the generic representation largely determines the range of possible outcomes. A tight representation based on previously near-optimal solutions may be fine for some engineering problems but might seriously inhibit the range of more creative solutions in another domain. For example, parameterization is a valuable technique for exploring variations on a well-tried and tested theme, but it is limited to types of variation that were anticipated when the parameterization was established. A very open representation is often difficult to imagine and can easily generate a vast search space.

The nature of the simplification in the genetic code for step two can also largely predetermine the range of possible solutions.

The "pioneer genes" in the third step of generating a population of code scripts are often known near-optimal solutions, but again this largely determines possible outcomes or again opens a vast search space.

The fourth and possibly most crucial step of epigenetic development in an environment is the least explored part of this process. Early approaches, particularly in engineering optimization, made this step redundant by having a direct mapping from the code script to the part or system under development. Others have favored a component-based approach (Bentley 2000). Processes closer to biological models of the embryogeny are also proposed (Frazer and Connor 1979; Bentley 1999).

For the fifth step of evaluation and selection, the criteria may be easy to define and apply for engineering optimization problems, but most design



9.3 FIGURE Yacht design by GA for a problem with conflicting and ill-defined criteria.

problems are classified as ill-defined (Simon 1969) or wicked (Buchanan 1990) or may have conflicting criteria. Interestingly, many selection techniques such as classifier systems can still perform well under these conditions, see Figure 9.3.

For the sixth step, crossover and mutation are the natural processes, but in our unnatural model other transformations may be useful.

And at the end even stopping turns out to be also highly problematic since it is never clear whether a much better design is about to be evolved in the next evolutionary cycle. Natural evolution is a continuous process consisting only of experiments—not the frozen moment that we need for design—but maybe even this concept will change.

9.5 CONCEPT SEEDING APPROACH

Our first attempts to overcome some of the problems outlined above were referred to as *concept seeding* (Frazer 1974, Frazer and Connor 1979). This approach required the designers to capture their design concepts in generic form.

Most designers employ a methodology that is highly personalized yet that can often be generic when the designer's body of work is taken as a whole. It is part of their working method and hence characterizes their "style" by which they are known. With an artist this style may be a clearly recognizable graphic technique, as with, say, the drawings of Beardsley; a painting technique, such as the impressionists; or perhaps a distinctive choice of palette, say, Titian. With architects and designers, the same is true. In some cases the style is also immediately recognizable from visual clues, such as the work of Gaudi or Mackintosh. But often the style is more organizational or procedural or concerned with more abstract space and form, such as with the work of Frank Lloyd Wright, although even with Wright the large roofs and low eaves of his early houses are an immediate stylistic giveaway.

This personalized but generic methodology can be described as a *design schema* in that it is an abstract conception of what is common to all designs. Inside the designer's office, these implicit *design schemas* often become formalized. It is common to find sets of standard details in architects' offices that serve to economize in time, ensure details are well tested, but also to ensure a consistency of detailing and to reinforce the house style. In many offices this extends to design procedures, approaches to organization, and so forth. The same is true of industrial designers, where again stylistic characteristics such as details, color, and preferred materials give economy, consistency, quality control, and identifiable house style. Again the design office may have other characteristics that make it identifiable, such as concentration on certain building types or product markets, or it may be more abstract characteristics, such as particular interest in social or user-centered concerns. Even these more abstract concepts can make the designer immediately recognizable, at least to fellow professionals, and often to the interested public. These characteristics even extend into areas of fashion, jewelry, and accessories, where particular designer labels are immediately recognizable on the street. The identifying characteristics often go through changes during the development of the designers, sometimes with abrupt changes as with Le Corbusier, but usually a continuous progression can be seen. The stylistic characteristics can continue with an office, studio, or company, long after the death of the original designer.

The evolutionary paradigm requires that this conceptual approach is captured and codified. This can be done historically by analysis as manifest by endless attempts to recapture the nature of the paintings of Mondrian or the villas of Palladio. The success varies depending on the sensitivity of the re-creator. Often the results are crass, particularly in the case of trying to create new works by dead artists. Mondrian would be horrified to see what some programmers might think would pass as his work. Often the help of the living designer can be involved. In the case of our work with Cedric Price and Walter Segal, the architects were alive and were involved in the capture process. Walter Segal is now dead, but we still have the design schema for his timber housing system captured for perpetuity in the software (Frazer 1982).

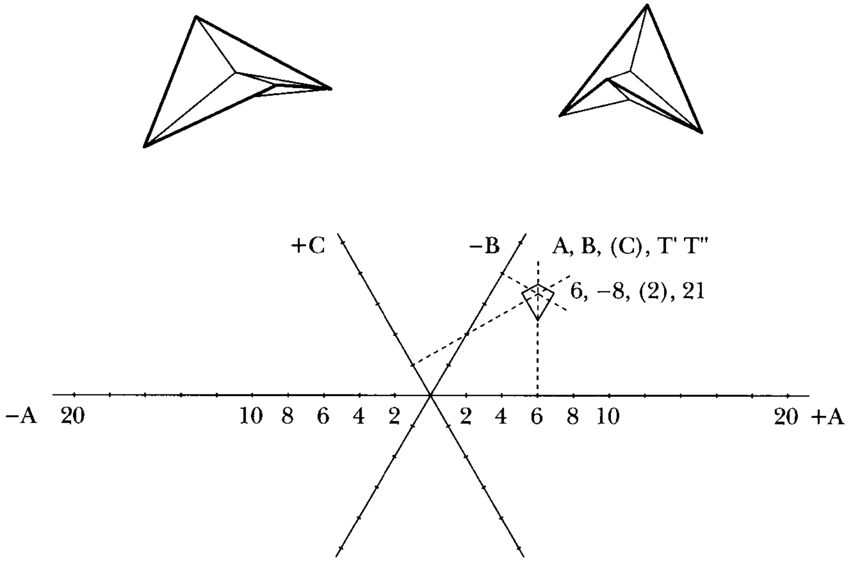
9.6 THE REPTILE DEMONSTRATION

I first attempted to realize this approach from 1968 onward with the *Reptile system.* This was an essentially component-based approach with epigenetic rules specific to the system.

The first computer drawings of this project were in 1967 at the Architectural Association, but they did not employ the seeding technique. In fact the difficulty of inputting data at that time was the main impetus to developing the seeding approach.

The first program was written in 1971 for the Reptile system. This system consists of two structural units of folded plate construction that can be orientated in 18 different ways relative to each other, resulting in many combinatorial possibilities. The combinations allow the system to produce enclosures of a very wide variety of plan shapes and complex structural forms.

Units are located by 60° axes using integer coordinates (A, B, C). "B" coordinates are negative in what one would conventionally expect to be a positive direction so that "C" coordinates (which are not stored) can be easily found ($A + B + C = 0$) and to provide a check when inputting data manually. A "D" coordinate describes depth or level of the unit in the structure. The type and orientation of a unit is described by two digits, the first giving the type and vertical orientation, the second the orientation in the horizontal plane ($T'T''$). This whole description of the unit in space takes the form (A, B, C, D, $T'T''$), for example, 66, 32, 34, 21, and was originally packed into one word of the Cambridge Atlas Titan with a pointer to the next unit in the chain. This we define as the genetic code script (Figure 9.4).



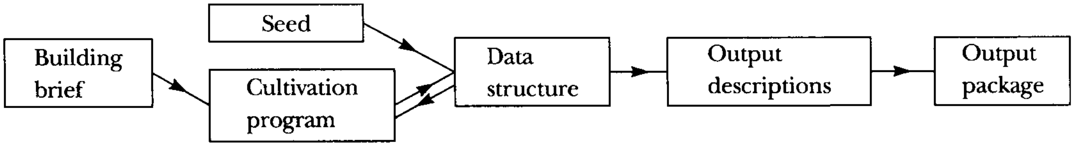
9.4 FIGURE The structural units and their representation in the genetic code script.

The basic data structure is set up and manipulated by a series of machine code subroutines and functions. These allow the descriptions of the units to be retrieved, deleted, or updated, and additional units to be inserted into the chain. On output the integer location on the 60° grid is translated into normal orthogonal coordinates, and the type and orientation of the unit call a subroutine that contains a description of the unit. A variety of descriptions are available depending on the detail required in the final drawing.

The seed is a minimal closed configuration of the units that includes units in all possible orientations but not necessarily all possible combinations. The development of a building is initialized by seeding the data structure with a description of the units making up the selected seeds with the units chained from the top of the seed downward in a clockwise direction.

This initial seeded data structure is then manipulated by a series of Fortran subroutines that enable the seed to be grown, stretched, and sheared, until the required building form is produced; see Figure 9.5 (Frazer 1974).

Much of the development work on this program had to be done at night in the Cambridge University Mathematical Laboratory using the then only graphics terminal in Cambridge driven by a dedicated PDP7 from the Atlas Titan. The



9.5 FIGURE The epigenetic process.

other main user of the graphics system, with whom I had to compete for booking time, was John Horton Conway, who was using the system to develop the Life game. Although little explicit connection was made at the time, the relevance of his work to the development of this project is obvious.

The particular configuration of the units in the seed has a significant effect on the final overall building form. The information about the type and orientation of the starting unit and those adjacent to it in the data chain will vary from seed to seed and this will affect the choice in the program of procedure for infilling units in the data chain (Figure 9.6). This is analogous to the difference between the transition rules in cellular automata and the initial configuration (often referred to as the seed).

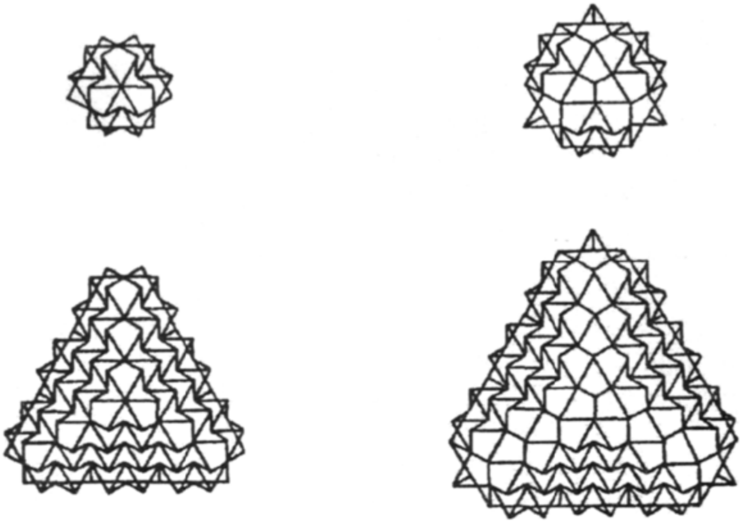
In this early and restricted version of the program only two seeds were ever used, the knot (containing 42 units) and the star (containing 72 units). An example of a building plan generated from the star seed is shown in Figure 9.7.

At the time when the epigenetic Reptile program was written, ideas of genetic algorithms were just developing. If I had tried using GAs, could I have evolved different seeds? I had painstakingly designed two by hand, and one of my students later developed a third.

Instead a generalized, but still component-based, version of the program was developed. The program required two kinds of information: first, the conceptual model of the building information in its minimal coded configuration and, second, a description of the actual components and details for the output stage.

The item description in the seed is not necessarily a building component but is more usually an assembly of components in a key configuration such as a corner or change of direction. The item is again located by just one point, and this point is usually taken as a change in surface direction to facilitate daylight and similar calculations.

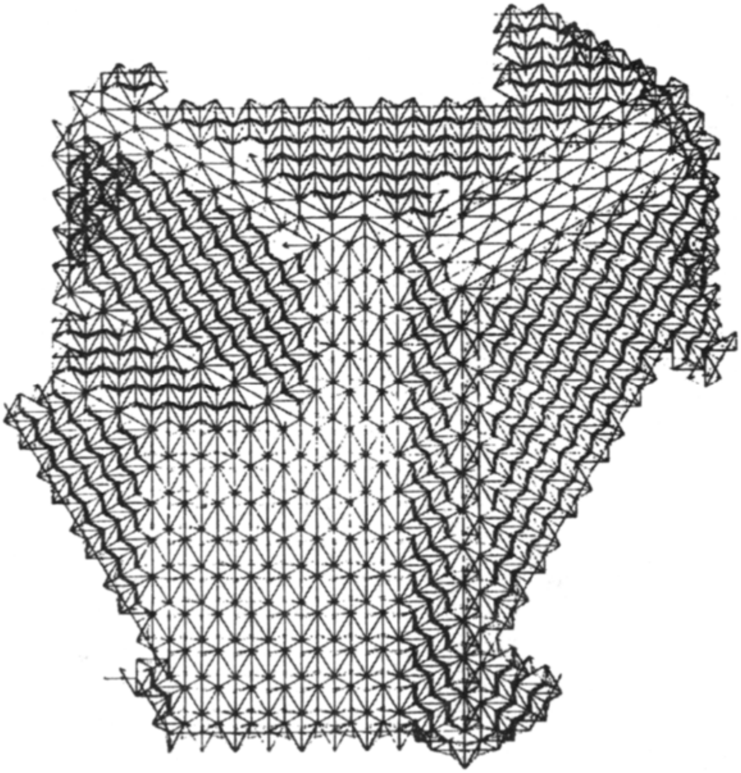
The concept of cultivating the seed to produce different buildings has been extended to the concept of mutating details of the seed to produce variants to overcome the great increase in environmental variety encountered with more general-purpose building construction. Individual mutations of the assemblies



9.6

FIGURE

Building forms growing from two different seeds during the epigenetic process.



9.7

FIGURE

Plan of building generated from star seed.

can be developed interactively and stored as variants that are referred to by an additional set of digits in the item description on the data chain.

As with the Reptile program, all the necessary information for the creation of new items in the data chain is derived from the type, location, and orientation of the item in the seed and those adjacent to it in the chain at the point where a move or change is to be made. An important feature of the program is that it only takes account of dimensional coordination if this is essential to the concept. It does not rely on modular coordination or a grid but computes the distances between locating points of items in the chain, to determine modules appropriate to the construction being considered.

In operation the configuration of the minimal construction or seed is compared with the user's requirements for a particular building, and the seed is grown, stretched, deformed, and pruned until it conforms with these requirements. For the same set of building requirements, many forms of building may be possible even from the same seed, and also the same form of building can be produced by several different cultivation routines. The seed is compared with the brief for the building, and the data structure is modified in two ways. First, the quantifiable and specific parts of the brief are dealt with as far as possible in the form of optimization routines. Alternative strategies for cultivating the seed are automatically evaluated, and the program adjusts itself on a simple heuristic basis to adopt those tactics that have proved most successful in previous attempts. Second, the user evaluates a series of solutions produced by the first technique in terms of nonquantifiable criteria, including aesthetic judgments. The program can either learn to adapt to the aesthetic prejudices of a particular user, or the types of solution can be classified in terms of formal or aesthetic characteristics that could then be requested by the user (Frazer and Connor 1979).

9.7 UNIVERSAL STATE SPACE MODELER

The first step was to establish a suitable computer environment for our model. We call this architectural genetic search space a "Universal State Space Modeler" or "isospatial model." "Universal" because it can model any space or structure or environment equally; "isospatial" because it is isometric, isomeric, isomorphic, and isotropic. The geometry of the space is also recursively self-similar and can be termed "homospatial," as it is a fractal, and therefore scaleless, space. This recursive property allows hierarchies of complexity to be described in the same logical space (Frazer 1992).

Other properties are made evident by the name of the model. We use the term "state space" because each logical point in the space can exist in many different states; each point in the state space corresponds to a virtual state of a system being modeled.

A *mote* is a point in a regular geometrical array that carries with it information about where it is, and to which other motes (close or remote) it is connected in some relational sense; it has information on its own properties including a parameterized geometrical description of any form to which it might map, (most importantly) a history of how it got to be where and what it is, and finally a set of rules concerned with why it is what and where it is. The mote itself does not move, but its identity or properties may move in response to some transformational activity. The field of motes extends indefinitely, and a particular object (structure) defined by a group of motes is seen as surrounded by motes that can also have properties and model the environment. Thus the behavior and performance of a proposed structure can be modeled in an environment. More significantly, the form of the structure may respond to an environment in a manner analogous to a genotype producing a phenotypic reaction. The rules of the mote may also be allowed to evolve in response to many iterative interactions with the environment.

The mote is a minute particle in a regular geometrical configuration. An orthogonal array is possible and has the advantage of easy representation, algorithms, and visualization as it can be considered like an array of voxels or cells in a spatial enumeration system. This is also the main disadvantage because it might lead to misunderstandings about the nature of the mote, and it would also share some of the disadvantages, in particular, appearing to have a geometrical bias and having nonequivalent neighbors. (That is, a cell in a cubic array has 6 face neighbors, 12 edge neighbors, and 8 vertex neighbors. They are of three different spatial and geometric relationships and three different center distances). Also 26 neighbors is a lot of neighbors to talk to! This problem can be partly avoided by considering face neighbors only and communicating with edge neighbors via a face neighbor, and with a vertex neighbor via a face neighbor and its face neighbor. This route of communication is complex.

An alternative (and currently preferred) geometry uses the centers of close packing spheres (cubic face centered packing). Each mote thus has 12 neighbors of identical geometric conformation and equal center-to-center distance. The reduction in number, equidistance between centers, and ease of identification in the database are all significant advantages for certain applications. The close packing configuration can also be oriented as a whole to allow easy visualization and assimilation of orthogonal structures, but it can also be oriented with closest packing layers horizontal.

In this case, considering the system as hexagonal layers, a coordinate system based on four axes at mutual angles of 109 degrees 28 minutes is most convenient. The location of any point in the spatial array is then given by the intersection of the four planes perpendicular to the axes. Labeling the positive coordinate directions as those most mutually opposed produces the useful relationship that the sum of the four coordinates of a point is always zero (i.e., if the four axes are labeled p, q, r, s, then $p + q + r + s = 0$). Thus only the most convenient three axes need be used for a particular point, or all four can be used and the relationship used as a check. This technique is based closely on the previous Reptile system.

In the case of the cubic orientation, a more normal three-coordinate Cartesian x, y, z system will suffice but is less elegant.

The two systems will often usefully be used together. The relationship between the coordinate systems is given by

$$x = (p + r)/k \qquad y = (q + r)/k \qquad z = (p + q)/k$$

or

$$\begin{array}{ll} p = (x + y + z)k/2 & q = (-x - y + z)k/2 \\ r = (x - y - z)k/2 & s = (-x + y - z)k/2 \end{array}$$

Using the four-coordinate system described above, the 12 adjacent motes are simply given by

$$\begin{array}{llll} (+1, 0, -1, 0) & (+1, 0, 0, -1) & (+1, -1, 0, 0) & (0, +1, -1, 0) \\ (0, +1, 0, -1) & (0, 0, +1, -1) & (0, -1, +1, 0) & (0, -1, 0, +1) \\ (0, 0, -1, +1) & (-1, 0, 0, +1) & (-1, +1, 0, 0) & (-1, 0, +1, 0) \end{array}$$

These conveniently represent all possible permutations that satisfy $p + q + r + s = 0$, thus ensuring no redundancy or waste in the packing of the data structure.

Furthermore, the entire array is recursively subdividable. Consider a configuration of the points in the array defining a tetrahedron. Then the tetrahedron can be repeatedly subdivided by any convenient multiple as a self-similar array. This fractal characteristic produces significant data storage economies.

9.8 LOGIC FIELDS

Every mote is probably best considered as being analogous to a piece of object-oriented code. It carries a complete copy of all instructions needed for the generation of forms of a particular species, although for programming economy these are not actually stored more than once. The important point is that there is not one master mote with all the key instructions, nor is there a master controlling program outside the data structure. The data structure of motes is the program in the sense that only the whole knows about the whole. Specific instances are manifestations of a response to outside stimuli or environment. All this is broadly analogous to each cell in the human body having a complete copy of the DNA and producing specific forms (phenotypic reactions) in response to a particular environment.

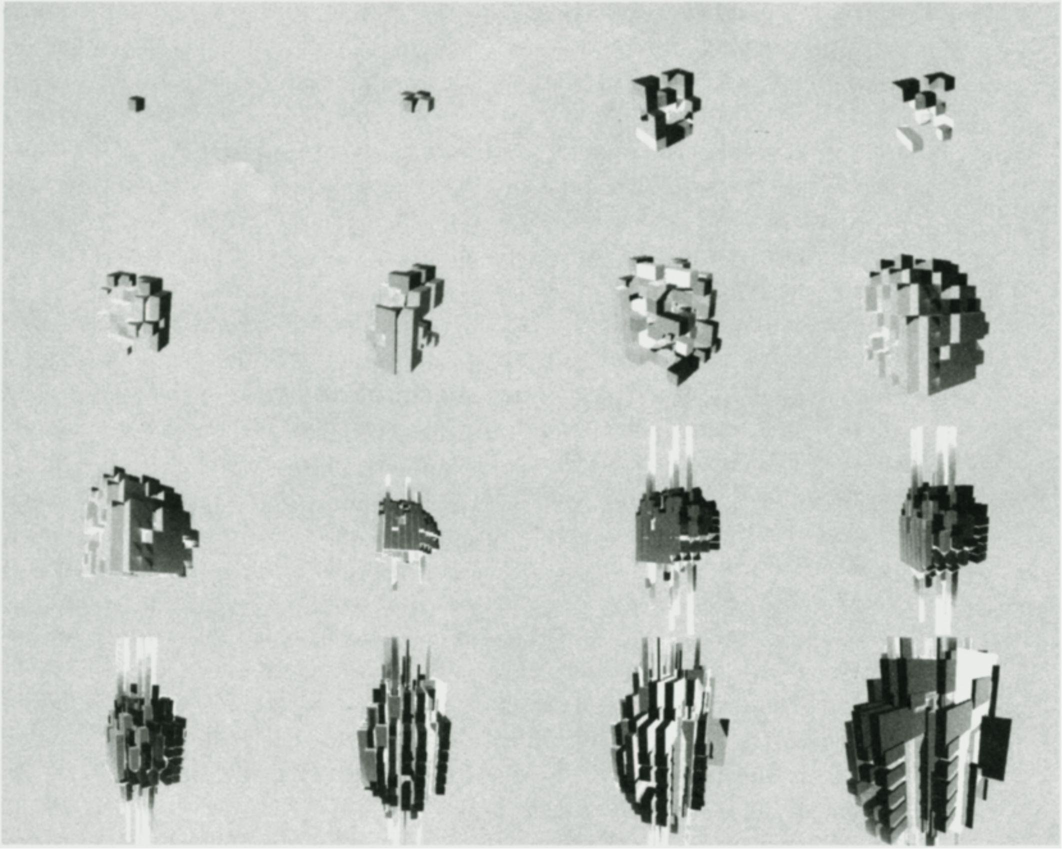
Each mote knows where it is (i.e., its geometrical relationship to its neighbors), and it knows the identity of its neighbors. It also knows its own identity, which might include properties. But most significantly, it knows why it is what it is where it is. It has this understanding by knowing a history of how it came to be where it is and what rules led to this happening and the nature of any external stimuli that were involved in this process. It is thus able to reproduce the development of a specific form. More importantly, if part of a form changes, it is able to know if it must also change. This occurs by passing messages from neighbor to neighbor, agreeing on a course of action, and then moving synchronously.

In this sense it is like cellular automata. Like cellular automata, the state of a mote at time t is dependent on its state and its neighbors' state at time $t - 1$ along with transition rules. The mote is similar except that its neighbors may include motes that are external to the structure (i.e., environment) and that neighbors (including environment) may be very remote from the mote in question.

Within an evolutionary system, such cellular-automata-like growth processes are a possible technique for processes of epigenetic development. Thus the epigenetic development that leads from code scripts to the virtual model is essentially process driven. It no longer presupposes a set of components (as the Reptile system does) but a generative set of processes capable of producing the emergent forms, structures, and spaces of the models as a by-product of the cellular activity. One example of such an epigenetic development process maps the states of each cell to geometrical transformations that deform the actual cell (Frazer 1995); see Figure 9.8.

Broadly speaking information travels through the array in the form of logic fields, by which means the form as a whole can go through an epigenetic development process in response to the environment.

Furthermore, it can learn which rules were successful in developing and modifying form and evolve new rules for improving its form-making ability. Usually this occurs in conceptual form by generating a computer model, which may then be manufactured as a frozen moment in the evolutionary process. In



9.8 FIGURE Genetic algorithm with multistate three-dimensional cellular automata as a process of epigenetic development. The growth of the cells is controlled by transition rules and the states of the cells.

special cases the system may go on to exhibit this behavior after manufacture. In this case, not only is the conceptual model "intelligent," but the artifact or structure might also be said to be "intelligent."

We thus have a concept of the design process as a computer evolving and modeling its own structure in response to environmental influences that are part of the same field. Just as the structure is influenced by the environment, so too does the influence of the structure extend beyond its own boundaries. A particular manufactured or built object in the real world is just a frozen moment in this process. In special cases, a manufactured or built object may continue to respond to environmental changes after construction and manufacture, and indeed it may be an increasingly normal part of our environment to find buildings and artifacts continuing to change and develop in response to a changing environment.

The genetic approach to design attempts to address the problem by formulating a more fundamental description of the generic characteristics of a design approach that then can be adapted to a specific design problem. The assertion is that this leads more naturally to the creative use of computer techniques and offers further possibilities for evolutionary design where many iterations of a design are used to gradually refine the design, or even automatic design where environmental constraints are introduced and specific designs produced automatically. These techniques can be combined to evolve specific designs in response to an environment and with feedback from a simulated environment, or better still, in response to the real environment. The environment might include potential users or the designer, who could then participate in its evolution.

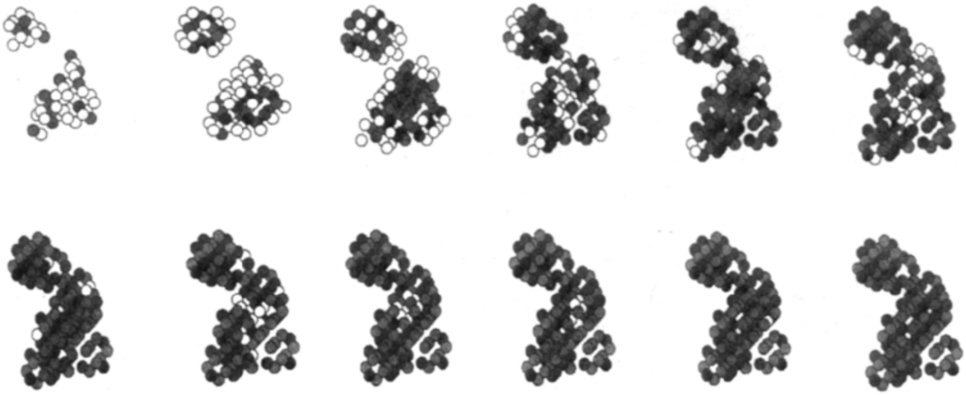
This leads to the concept of a designer creating genes that contain the rules for generating form (or phenotypic reaction) in response to an environmental stimulus. The computer may also be able to model and predict the evolutionary development in an accelerated manner and learn on the basis of its experience.

9.9 RETURNING TO THE ANALOGY WITH NATURE

The epigenetic process itself has evolved, the environment has (co-)evolved, other species have coevolved. Presumably the very process of evolution has evolved.

It is thus tempting to speculate to what extent we could evolve all the parts of our evolutionary design model. We have successfully evolved the epigenetic rules for forming features such as slabs and columns (Frazer 1995; Graham 1995). We have experimented with evolving both the seeds and the transition rules in a three-dimensional universe of close packing spheres (Frazer 1995; Graham 1995); see Figures 9.9 and 9.10. Our attempt to coevolve a structure with the environment using hierarchical cellular automata to also develop the rules was not successful the first time (Frazer 1995). We have demonstrated the cooperative evolution of cities with a project in Groningen (Frazer 1997). Perhaps we can evolve selection criteria?

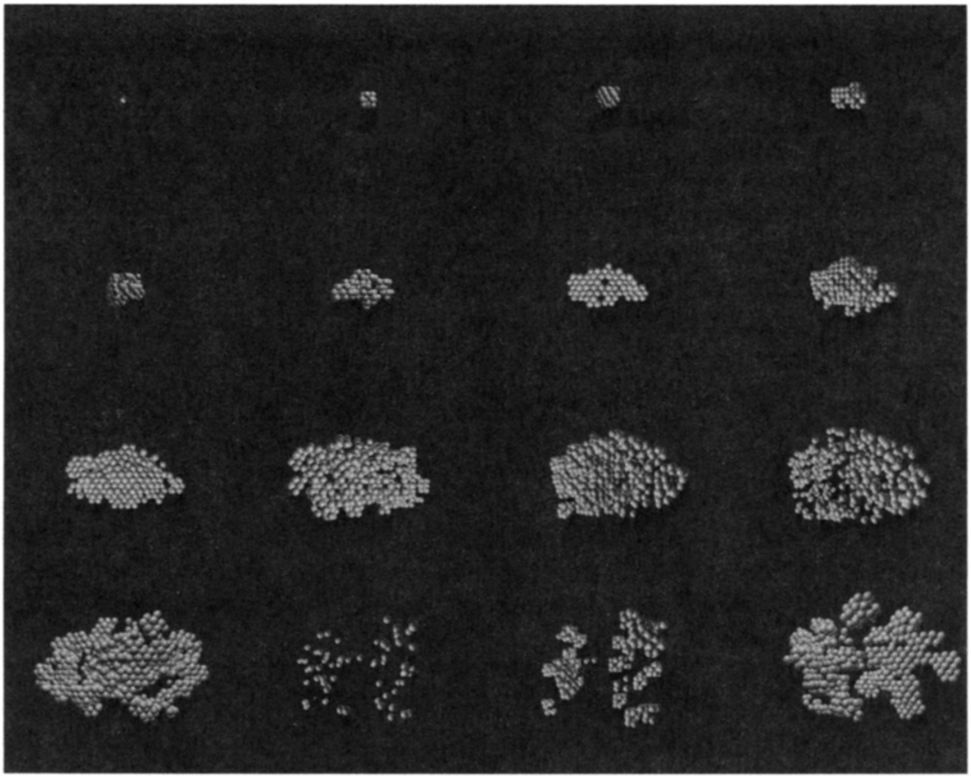
But perhaps we could go further. The next logical evolutionary step would be to evolve the seeds as well. Would this be possible? Could one set in motion a



9.9

FIGURE

Experiment to attempt to evolve artificial life out of nothing, with neither rules nor seed being given in the first instance. A genetic algorithm selected and developed successful rules and seeds. In this case success was the ability to survive and develop complex structures (Frazer 1995; Graham 1995).



9.10

FIGURE

Interactivator: Experimental cooperative evolution of form by interaction with visitors and the environment in an exhibition and via the Internet. Isospatial data structures with multistate data cells and control by nested genetic algorithms. Sequence shows the symmetry of the data cells breaking (Janssen 2000; Graham 1995; Frazer 1995).

process of creation that required massive feedback but only one seed? Darwinian evolutionary theory, after all, implies one common root for all species. Perhaps we do not even need one seed? Could we evolve everything from a form of primeval data soup, evolving not just the rules but even the starting configuration? Could it be possible to coevolve the very environment in which the epigenetic development will occur?

If a complete evolutionary model could be demonstrated of even a very simple design problem, then further light might be shed on the mysteries of the natural evolutionary process.

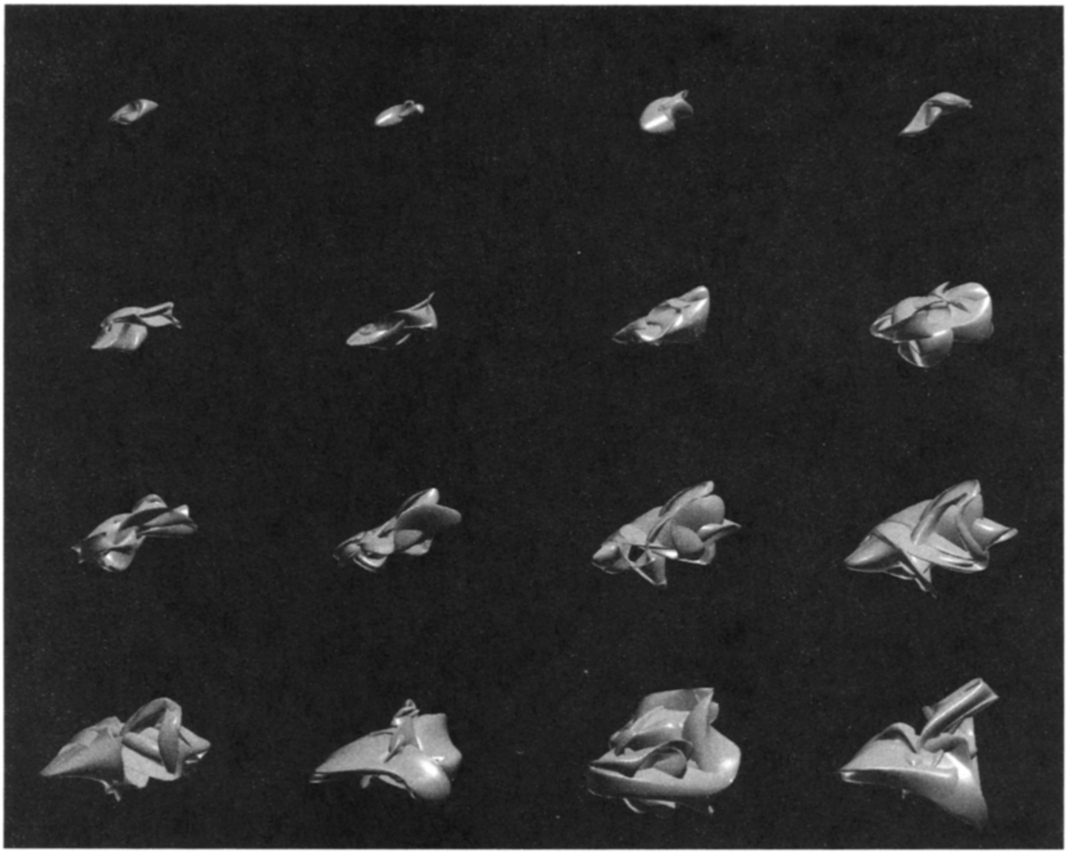
But perhaps it is wrong to make a differentiation between the evolution of the natural and the artificial? After all, we are part of the natural evolutionary process. I believe we may be moving toward hybrid systems of natural and artificial evolution of both natural and artificial things. Genetic engineering and its converse, architectural genetics, will converge or possibly even cross over as the virtual and actual worlds are converging and may cross, see Figure 9.11.

9.10 CONCLUSIONS

DNA does not describe the phenotype, but constitutes instructions that describe the process of building the phenotype, including instructions for making all the materials, then processing and assembling them. This includes making enzymes for the production of nucleotides, plus instructions for cell division and differentiation. These are all responsive to the environment as it proceeds, capable of modifying in response to conditions such as the availability of foodstuffs, and so on. Genes are not for shapes but for chemistry, and by analogy our model also describes process rather than form. This procedure is environmentally sensitive. The rules are constant, but the outcome varies according to materials or environmental conditions. Eventually it is our intention that the form-making process will be part of the system, but for the moment our model works by describing the processing and assembling of the materials. The actual processing and assembly is external to the model.

Simply stated, what we are evolving are the rules for generating form, rather than the forms themselves. We are describing processes, not components; ours is the packet-of-seeds as opposed to the bag-of-bricks approach.

What we have been talking about is machine-assisted human creativity. However surprising or interesting some emergent forms might be, the primary credit for creativity must still go to the originator of the seed or method or process. The



9.11
FIGURE

Experimental cooperative evolution of form by interaction with visitors and the environment in an exhibition and via the Internet. Isospatial data structures with multistate data cells and control by nested genetic algorithms. Sequence shows the visualization of the evolution of the resulting form (Janssen 2000; Graham 1995; Frazer 1995).

design of a building is attributed to the principal of the practice, with perhaps a mention of the job architect. So we are still at the stage where the design by computers is attributed to the originator of the method. We are not yet at the stage where the machine might be credited with being even the job architect, but we are making progress in that direction. Even if we could evolve the seed and the epigenetic process ab initio, there would still be a tendency for humans to credit the human who conceived of the system with the creativity. After all, even the inventions of Mother Nature are often credited to another.

REFERENCES

Bentley, P. J. (ed.) (1999). *Evolutionary Design by Computers.* Morgan Kaufmann.

Bentley, P. J. (2000). Exploring Component-Based Representations—The Secret of Creativity by Evolution? In the *Fourth International Conference on Adaptive Computing in Design and Manufacture* (ACDM 2000), April 26–28, 2000, University of Plymouth, UK.

Buchanan, R. (1990). Wicked Problems in Design Thinking. In V. Margolin and R. Buchanan (eds.), *The Idea of Design,* MIT Press, 1995.

Dawkins, R. (1983). *The Extended Phenotype.* Oxford University Press.

Dawkins, R. (1986). *The Blind Watchmaker.* Longman.

Dennett, D. (1995). *Darwin's Dangerous Idea, Evolution and the Meaning of Life.* Penguin Press.

Frazer, J. (1974). Reptiles. *Architectural Design.* (April):231–239.

Frazer, J. (1982). Use of Simplified Three-Dimensional Computer Input Devices to Encourage Public Participation in Design. *Computer Aided Design* 82, conference proceedings, Butterworth Scientific, pp. 143–151.

Frazer, J. (1992). Datastructures for Rule-Based and Genetic Design. In T. L. Kunii (ed.), *Visual Computing—Integrating Computer Graphics with Computer Vision,* Springer-Verlag, pp. 731–744.

Frazer, J. (1994). The Genetic Language of Design. In S. Braddock and M. O'Mahoney (eds.), *Textiles and New Technology: 2010,* Artemis, pp. 77–79.

Frazer, J. (1995). *An Evolutionary Architecture.* Architectural Association Publications.

Frazer, J. (1997). Action and Observation: The Groningen Experiment. *Abstracts of Papers for Problems of Action and Observation Conference,* Amsterdam, April 1997, pp. 14–16.

Frazer, J., and J. Connor (1979). A Conceptual Seeding Technique for Architectural Design. In *Proceedings of International Conference on the Application of Computers in Architectural Design,* Berlin, Online Conferences with AMK, pp. 425–34.

Frazer, J., M. Rastogi, and P. Graham (1995). Biodiversity in Design via Internet. *Digital Creativity, A Conference on Computers in Art & Design Education,* CADE 95, Brighton, pp. 97–106.

Gould, S. (1989). *Wonderful Life: The Burgess Shale and the Nature of History.* Norton.

Graham, P. (1995). *Evolutionary and Rule Based Techniques in Computer Aided Design.* Doctorate thesis, University of Ulster.

Graham, P., J. Frazer, and M. Hull (1993). The Application of Genetic Algorithms to Design Problems with Ill-Defined or Conflicting Criteria. In R. Glanville and G. de Zeeuw (eds.), *Proceedings of Conference on Values and (In) Variants,* Amsterdam, pp. 61–75.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Janssen, P. (2000). Search, Search Spaces and Fitness Landscapes in Design: A Conceptual Reappraisal with Implications for Generative Systems. In *Proceedings of Artificial Intelligence in Design 2000,* Kluwer Academic (in press).

Simon, H. (1969). *The Sciences of the Artificial.* MIT Press.

Sims, K. (1991). Artificial Evolution for Computer Graphics. *Computer Graphics* 25(4).

10

CHAPTER

Genetic Programming: Biologically Inspired Computation That Exhibits Creativity in Producing Human-Competitive Results

John R. Koza Stanford University
Forrest H. Bennett III Fuji Xerox Palo Alto Laboratories
David Andre University of California, Berkeley
Martin A. Keane Econometrics, Inc.

10.1 INTRODUCTION

One of the central challenges of computer science is to get a computer to solve a problem without programming it explicitly. In particular, the challenge is to create an automatic system whose input is a high-level statement of a problem's requirements and whose output is a satisfactory solution to the given problem. This challenge is the common goal of such fields of research as artificial intelligence and machine learning. Paraphrasing Arthur Samuel (1959), this challenge addresses the question: How can computers be made to do what needs to be done, without being told exactly how to do it? As Samuel further explained: "The aim [is] . . . to get machines to exhibit behavior, which if done by humans, would be assumed to involve the use of intelligence" (Samuel 1983).

This chapter provides an affirmative answer to the following two questions: Starting only with a high-level statement of the problem's requirements, can computers automatically discover the solution to nontrivial problems? And, can automatically created solutions be competitive with the products of human creativity and inventiveness?

In answering these questions, this chapter focuses on a biologically inspired domain-independent problem-solving technique of evolutionary computation, called genetic programming. The technique of genetic programming is applied to three problems for which there is no known general mathematical solution:

- Automatically synthesizing (designing) both the topology (graphical arrangement of components) and sizing (component values) for an analog electrical circuit
- Automatically synthesizing the placement and routing of electrical components (while simultaneously synthesizing the topology and sizing)
- Automatically synthesizing both the topology and tuning (component values) for a controller

For each problem, genetic programming automatically creates entities that improve on previously patented inventions, or duplicate the functionality of previously patented inventions.

Section 10.2 states what we mean when we say that an automatically created solution to a problem is competitive with the product of human creativity. In Section 10.3 we briefly describe genetic programming, building on what was already explained in the introduction to this book. In Section 10.4 we discuss the problem of automatically synthesizing both the topology and sizing for an analog electrical circuit. In Section 10.5 we discuss the problem of automatically determining the placement and routing (while simultaneously synthesizing the topology and sizing) of an electrical circuit. Section 10.6 discusses the problem of automatically synthesizing both the topology and tuning for a controller. Finally, Section 10.7 discusses the importance of illogic in achieving creativity and inventiveness.

10.2 INVENTIVENESS AND CREATIVITY

What do we mean when we say that an automatically created solution to a problem is competitive with the product of human creativity and inventiveness?

We are not referring to the fact that a computer can rapidly print ten thousand payroll checks or that a computer can compute π to a million decimal places. As Fogel, Owens, and Walsh (1966) said, "Artificial intelligence is realized only if an inanimate machine can solve problems . . . not because of the machine's sheer speed and accuracy, but because it can discover for itself new techniques for solving the problem at hand."

We think it is fair to say that an automatically created result is competitive with one produced by human engineers, designers, mathematicians, or programmers if it satisfies any of the following eight criteria (or any other similarly stringent criterion):

A. The result was patented as an invention in the past, is an improvement over a patented invention, or would qualify today as a patentable new invention.

B. The result is equal to or better than a result that was accepted as a new scientific result at the time when it was published in a peer-reviewed scientific journal.

C. The result is equal to or better than a result that was placed into a database or archive of results maintained by an internationally recognized panel of scientific experts.

D. The result is publishable in its own right as a new scientific result (independent of the fact that the result was mechanically created).

E. The result is equal to or better than the most recent human-created solution to a long-standing problem for which there has been a succession of increasingly better human-created solutions.

F. The result is equal to or better than a result that was considered an achievement in its field at the time it was first discovered.

G. The result solves a problem of indisputable difficulty in its field.

H. The result holds its own or wins a regulated competition involving human contestants (in the form of either live human players or human-written computer programs).

Note that each of the above criteria are couched in terms of *producing results* and also that the results are measured in terms of standards that are entirely external to the fields of artificial intelligence and machine learning.

Using the above criteria, there are now at least 25 instances where genetic programming has produced a result that is competitive with human performance. These examples come from fields such as quantum computing, the annual Robo Cup competition, cellular automata, computational molecular biology, sorting networks, the automatic synthesis of the design of analog electrical circuits, and the automatic synthesis of the design of controllers.

Table 10.1 shows 25 instances of results where genetic programming has produced results that are competitive with the products of human creativity and

#	Claimed instance	Basis for claim	Reference	Infringed patent
1	Creation, using genetic programming, of a better-than-classical quantum algorithm for the Deutsch-Jozsa "early promise" problem	B, F	(Spector, Barnum, and Bernstein 1998)	
2	Creation, using genetic programming, of a better-than-classical quantum algorithm for the Grover's database search problem	B, F	(Spector, Barnum, and Bernstein 1999)	
3	Creation, using genetic programming, of a quantum algorithm for the depth-2 AND/OR query problem that is better than any previously published result	B, D	(Spector et al.)	
4	Creation of soccer-playing program that ranked in the middle of the field of 34 human-written programs in the Robo Cup 1998 competition	H	(Andre and Teller 1998)	
5	Creation of four different algorithms for the transmembrane segment identification problem for proteins	B, E	(Koza et al. 1999)	
6	Creation of a sorting network (O'Connor and Nelson 1962) for seven items using only 16 steps	A, D	(Koza et al. 1999a)	
7	Rediscovery of the ladder topology for lowpass and highpass filters	A, F	(Koza et al. 1999a)	(Campbell 1917)
8	Rediscovery of "*M*-derived half section" and "constant *K*" filter sections	A, F	(Koza et al. 1999a)	(Zobel 1925)
9	Rediscovery of the Cauer (elliptic) topology for filters	A, F	(Koza et al. 1999a)	(Cauer 1934, 1935, 1936)
10	Automatic decomposition of the problem of synthesizing a crossover filter	A, F	(Koza et al. 1999a)	(Zobel 1925)
11	Rediscovery of a recognizable voltage gain stage and a Darlington emitter-follower section of an amplifier and other circuits	A, F	(Koza et al. 1999a)	(Darlington 1952)

10.1 TABLE Twenty-five instances where genetic programming has produced human-competitive results. *(continued)*

#	Claimed instance	Basis for claim	Reference	Infringed patent
12	Synthesis of 60 and 96 decibel amplifiers	A, F	(Koza et al. 1999a)	
13	Synthesis of analog computational circuits for squaring, cubing, square root, cube root, logarithm, and Gaussian functions	A, D, G	(Koza et al. 1999a)	
14	Synthesis of a real-time analog circuit for time-optimal control of a robot	G	(Koza et al. 1999a)	
15	Synthesis of an electronic thermometer	A, G	(Koza et al. 1999a)	
16	Synthesis of a voltage reference circuit	A, G	(Koza et al. 1999a)	
17	Creation of a cellular automata rule for the majority classification problem that is bet ter than the Gacs-Kurdyumov-Levin (GKL) rule and all other known rules written by humans	D, E	(Andre, Bennett, and Koza 1996)	
18	Creation of motifs that detect the D–E–A-D box family of proteins and the manganese superoxide dismutase family	C	(Koza et al. 1999a)	
19	Synthesis of analog circuit equivalent to Philbrick circuit	A	(Koza et al. 1999a)	
20	Synthesis of NAND circuit	A	(Koza et al. 1999a)	
21	Synthesis of digital-to-analog converter (DAC) circuit	A	(Bennett et al. 1999)	
22	Synthesis of analog-to-digital (ADC) circuit	A	(Bennett et al. 1999)	
23	Synthesis of topology, sizing, placement, and routing of analog electrical circuits	G	(Koza and Bennett 1999)	
24	Synthesis of topology for a PID type of controller	A, F	(Koza et al. 2000)	(Callender and Stevenson 1939)
25	Synthesis of topology for a controller with a second derivative	A, F	(Koza et al. 2000)	(Jones 1942)

inventiveness. Each claim is accompanied by the particular criterion (from the above list) that establishes the basis for the claim. As can be seen in the table, seven of these automatically created results would infringe on previously issued patents were those previous patents still in force. In addition, one of the genetically evolved results improves on a previously issued patent. Also, nine of the other genetically evolved results duplicate the functionality of previously patented inventions in a novel way. Since nature routinely uses evolution and natural selection to create designs for complex structures that are well adapted to their environments, it is not surprising that many of these examples involve the design of complex structures.

10.3 GENETIC PROGRAMMING

Genetic programming is an extension of the genetic algorithm described in John Holland's pioneering book *Adaptation in Natural and Artificial Systems* (Holland 1975). Genetic programming applies a genetic algorithm to search the space of possible computer programs.

Genetic programming progressively breeds a population of computer programs over a series of generations by starting with a primordial ooze of thousands of randomly created computer programs and using the Darwinian principle of natural selection, recombination (crossover), mutation, gene duplication, gene deletion, and certain mechanisms of developmental biology. Specifically, genetic programming starts with an initial population of randomly generated computer programs composed of the given primitive functions and terminals. The programs in the population are, in general, of different sizes and shapes. The creation of the initial random population is a blind random search of the space of computer programs composed of the problem's available functions and terminals.

On each generation of a run of genetic programming, each individual in the population of programs is evaluated as to its fitness in solving the problem at hand. The programs in generation 0 of a run almost always have exceedingly poor fitness for nontrivial problems of interest. Nonetheless, some individuals in a population will turn out to be somewhat more fit than others. These differences in performance are then exploited so as to direct the search into promising areas of the search space. The Darwinian principle of reproduction and survival of the fittest is used to probabilistically select, on the basis of fitness, individuals from the population to participate in various operations. A small percentage (e.g., 9%) of the selected individuals are reproduced (copied) from one generation to the next. A very small percentage (e.g., 1%) of the selected individuals are mutated in a random way. Mutation can be viewed as an undirected local search mechanism. The vast majority of the selected individuals (e.g., 90%) participate in the genetic operation of crossover (sexual recombination), in which two offspring programs are created by recombining genetic material from two parents.

The creation of the initial random population and the creation of offspring by the genetic operations are all performed so as to create syntactically valid, executable programs. After the genetic operations are performed on the current generation of the population, the population of offspring (i.e., the new generation) replaces the old generation. The tasks of measuring fitness, Darwinian selection, and genetic operations are then iteratively repeated over many generations. The computer program resulting from this simulated evolutionary process can be the solution to a given problem or a sequence of instructions for constructing the solution.

Probabilistic steps are pervasive in genetic programming. Probability is involved in the creation of the individuals in the initial population, the selection of individuals to participate in the genetic operations (e.g., reproduction, crossover, and mutation), and the selection of crossover and mutation points within parental programs.

The dynamic variability of the size and shape of the computer programs that are created during the run is an important feature of genetic programming. It is often difficult and unnatural to try to specify or restrict the size and shape of the eventual solution in advance.

Genetic programming is described in the book *Genetic Programming: On the Programming of Computers by Means of Natural Selection* (Koza 1992; Koza and Rice 1992), the book *Genetic Programming II: Automatic Discovery of Reusable Programs* (Koza 1994a, 1994b), and the book *Genetic Programming III: Darwinian Invention and Problem Solving* (Koza et al. 1999a, 1999b). Additional information on genetic programming can be found in the book by Banzhaf et al. (1998); in the series on genetic programming from Kluwer Academic Publishers (Langdon 1998); in edited collections of papers such as the Advances in Genetic Programming series of books from the MIT Press (Spector et al. 1999); in the proceedings of the Genetic Programming Conference (Koza et al. 1998); in the proceedings of the Euro-GP conference (Poli et al. 1999); in the proceedings of the Genetic and Evolutionary Computation Conference (Banzhaf et al. 1999); in the *Genetic Programming and Evolvable Machines* journal; and at Web sites such as *www.genetic-programming.org.*

The biological metaphor underlying genetic programming is very different from the underpinnings of most other techniques that have previously been tried in pursuit of the goal of automatically creating computer programs. Many computer scientists and mathematicians are baffled by the suggestion that biology might be relevant to solving important problems in their fields. However, we do not view biology as an unlikely source for a solution to the challenge of getting a computer to solve a problem without explicitly programming it. Quite the contrary—we view biology as a most likely source. Indeed, genetic programming is based on the only method that has ever produced intelligence—the time-tested method of evolution and natural selection.

Of course, the idea that machine intelligence may be realized using a biological approach is not a new one. Turing made the connection between searches and the challenge of getting a computer to solve a problem without explicitly programming it in his 1948 essay "Intelligent Machines": "Further research into intelligence of machinery will probably be very greatly concerned with 'searches' . . ." (Ince 1992). Turing then identified three broad approaches by which search might be used to automatically create an intelligent computer program.

One approach that Turing identified in 1948 is a search through the space of integers representing candidate computer programs. This approach, of course, uses many of the techniques that Turing used in his own work on the foundations of computation.

A second approach is the "cultural search," which relies on knowledge and expertise acquired over a period of years from others. This approach is akin to present-day knowledge-based systems.

The third approach that Turing specifically identified in 1948 is "genetical or evolutionary search." Turing said: "There is the genetical or evolutionary search by which a combination of genes is looked for, the criterion being the survival value. The remarkable success of this search confirms to some extent the idea that intellectual activity consists mainly of various kinds of search" (Turing, quoted in Ince 1992).

Turing did not specify how to conduct the "genetical or evolutionary search" for a computer program. However, Ince suggested how, in his 1950 paper "Computing Machinery and Intelligence," natural selection and evolution might both be incorporated into the search for intelligent machines:

> We cannot expect to find a good child-machine at the first attempt. One must experiment with teaching one such machine and see how well it learns. One can then try another and see if it is better or worse. There is an obvious connection between this process and evolution, by the identifications
> Structure of the child machine = Hereditary material
> Changes of the child machine = Mutations
> Natural selection = Judgment of the experimenter (Ince 1992)

Genetic programming implements Turing's third way to achieve machine intelligence.

10.4 APPLYING GENETIC PROGRAMMING TO CIRCUIT SYNTHESIS

The design process entails creation of a complex structure to satisfy user-defined requirements. The field of design is a good source of problems that can be used for determining whether an automated technique can produce results that are competitive with human-produced results. Design is a major activity of practicing engineers. Since the design process typically entails trade-offs between competing considerations, the end product of the process is usually a satisfactory and compliant design as opposed to a perfect design. Design is usually viewed as requiring creativity and human intelligence.

The *topology* of a circuit includes specifying the gross number of components in the circuit, the type of each component (e.g., a capacitor), and a *netlist* specifying where each lead of each component is to be connected. *Sizing* involves specifying the values (typically numerical) of each of the circuit's components.

The design process for analog electrical circuits begins with a high-level description of the circuit's desired behavior and characteristics and includes creation of the topology and sizing of a satisfactory circuit.

The field of design of analog and mixed analog-digital electrical circuits is especially challenging because (prior to genetic programming) there has been no previously known general technique for automatically creating the topology and sizing of an analog circuit from a high-level statement of the design goals of the circuit.

Although considerable progress has been made in automating the synthesis of certain categories of purely digital circuits, the synthesis of analog circuits has not proved to be as amenable to automation. As Aaserud and Nielsen (1995) observe:

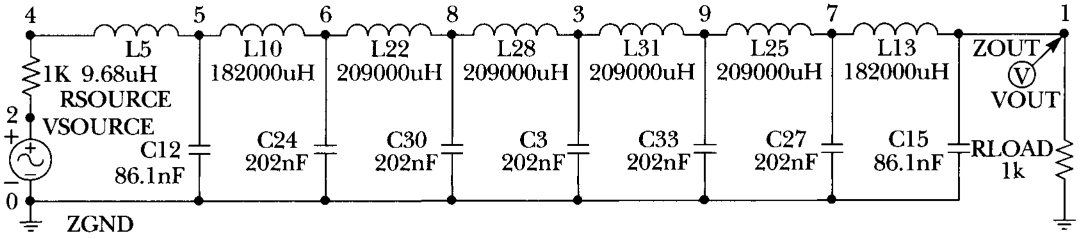
> Analog designers are few and far between. In contrast to digital design, most of the analog circuits are still handcrafted by the experts or so-called 'zahs' of analog design. The design process is characterized by a combination of experience and intuition and requires a thorough knowledge of the process characteristics and the detailed specifications of the actual product. Analog circuit design is known to be a knowledge-intensive, multiphase, iterative task, which usually stretches over a significant period of time and is performed by designers with a large portfolio of skills. It is therefore considered by many to be a form of art rather than a science.

We use filter circuits to demonstrate the automatic synthesis of analog electrical circuits using genetic programming. A simple analog *filter* is a one-input, one-output circuit that receives a signal as its input and passes the frequency components of the incoming signal that lie in a specified range (called the *passband*) while suppressing the frequency components that lie in all other frequency ranges (the *stopband*). Specifically, the goal is to design a lowpass filter composed of capacitors and inductors that passes all frequencies below 1,000 Hz and suppresses all frequencies above 2,000 Hz.

Genetic programming can be applied to the problem of synthesizing circuits if a mapping is established between the program trees (rooted, point-labeled trees with ordered branches) used in genetic programming and the labeled cyclic graphs germane to electrical circuits. The principles of developmental biology provide the motivation for mapping trees into circuits by means of a developmental process that begins with a simple embryo. For circuits, the initial circuit typically includes a test fixture consisting of certain fixed components (such as a source resistor, a load resistor, an input port, and an output port) as well as an embryo consisting of one or more modifiable wires. Until the modifiable wires are modified, the circuit does not produce interesting output. An electrical circuit is developed by progressively applying the functions in a circuit-constructing program tree to the modifiable wires of the embryo (and, during the developmental process, to succeeding modifiable wires and components). A single electrical circuit is created by executing the functions in an individual circuit-constructing program tree from the population. The functions are progressively applied in a developmental process to the embryo and its successors until all of the functions in the program tree are executed. That is, the functions in the circuit-constructing program tree progressively side-effect the embryo and its successors until a fully developed circuit eventually emerges. The functions are applied in a breadth-first order.

The functions in the circuit-constructing program trees are divided into five categories:

- Topology-modifying functions that alter the topology of a developing circuit
- Component-creating functions that insert components into a developing circuit
- Development-controlling functions that control the development process by which the embryo and its successors become a fully developed circuit
- Arithmetic-performing functions that appear in subtrees as argument(s) to the component-creating functions and specify the numerical value of the component
- Automatically defined functions that appear in the automatically defined functions and potentially enable certain substructures of the circuit to be reused (with parameterization)



10.1 FIGURE Evolved filter circuit that would infringe on Campbell's patent.

Before applying genetic programming to a problem of circuit design, seven major preparatory steps are required: (1) identify the embryonic circuit, (2) determine the architecture of the circuit-constructing program trees, (3) identify the primitive functions of the program trees, (4) identify the terminals of the program trees, (5) create the fitness measure, (6) choose control parameters for the run, and (7) determine the termination criterion and method of result designation. A detailed discussion concerning how to apply these seven preparatory steps to a particular problem of circuit synthesis (such as a lowpass filter) is found in Koza et al. (1999a), Chapter 25.

10.4.1 Campbell 1917 Ladder Filter Patent

The best circuit (Figure 10.1) of generation 49 of one run of genetic programming (Koza et al. 1999a) on the problem of synthesizing a lowpass filter is a 100% compliant circuit (i.e., it complies with all requirements for attenuation, passband ripple, and stopband ripple).

The evolved circuit is what is now called a cascade (ladder) of identical π sections (shown and analyzed in Koza et al. 1999a, Chapter 25). The evolved circuit has the recognizable topology of the circuit for which George Campbell of American Telephone and Telegraph received U.S. patent 1,227,113 in 1917. Claim 2 of Campbell's patent covered (Campbell 1917):

> An electric wave filter consisting of a connecting line of negligible attenuation composed of a plurality of sections, each section including a capacity element and an inductance element, one of said elements of each section being in series with the line and the other in shunt across the line, said capacity and inductance elements having precomputed values dependent upon the upper limiting frequency and the lower limiting frequency of a range of

> frequencies it is desired to transmit without attenuation, the values of said capacity and inductance elements being so proportioned that the structure transmits with practically negligible attenuation sinusoidal currents of all frequencies lying between said two limiting frequencies, while attenuating and approximately extinguishing currents of neighboring frequencies lying outside of said limiting frequencies.

In addition to possessing the topology of the Campbell filter, the numerical value of all the components in the evolved circuit closely approximate the numerical values specified in Campbell's 1917 patent. But for the fact that this 1917 patent has expired, the evolved circuit would infringe on the Campbell patent.

The legal criteria for obtaining a U.S. patent are that the proposed invention be "new" and "useful" and that ". . . the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would [not] have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains" (35 United States Code 103a).

Since filing for a patent entails the expenditure of a considerable amount of time and money, patents are generally sought, in the first place, only if an individual or business believes the inventions are likely to be useful in the real world and economically rewarding. Patents are only issued if an arm's-length examiner is convinced that the proposed invention is novel, useful, and satisfies the statutory test for unobviousness.

The fact that genetic programming rediscovered both the topology and sizing of an electrical circuit that was unobvious "to a person having ordinary skill in the art" establishes that this evolved result satisfies Arthur Samuel's criterion for artificial intelligence and machine learning (quoted in Section 10.1).

10.4.2 Zobel 1925 "*M*-Derived Half Section" Patent

Since the genetic algorithm is a probabilistic algorithm, different runs produce different results. In another run of this same problem of synthesizing a lowpass filter, a 100%-compliant circuit (Figure 10.2) was evolved in generation 34.

This evolved circuit (presented in Koza et al. 1999a, Chapter 25) is equivalent to a cascade of three symmetric T-sections and an *M*-derived half section. Otto Zobel of American Telephone and Telegraph Company invented and received a patent for an "*M*-derived half section" used in conjunction with one or more "constant K" sections. Again, the numerical value of all the components in

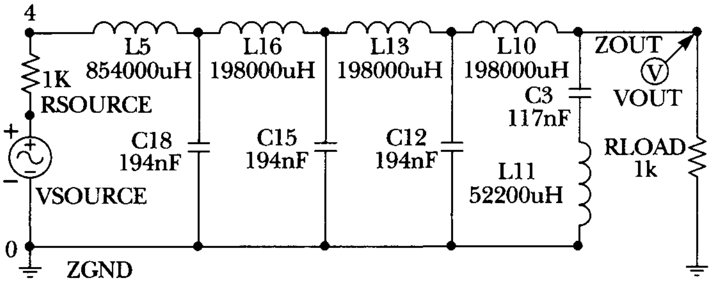


FIGURE 10.2 Evolved filter circuit that would infringe on Zobel's patent.

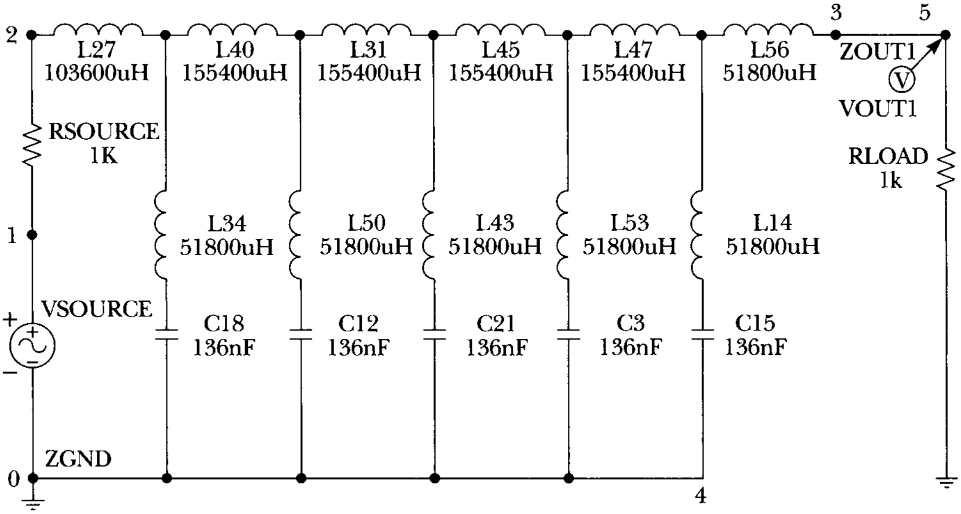
the evolved circuit closely approximate the numerical values specified in Zobel's 1925 patent.

10.4.3 Cauer 1934–1936 Elliptic Filter Patents

In yet another run of this same problem of synthesizing a lowpass filter, a 100% compliant circuit (Figure 10.3) emerged in generation 31 (Koza et al. 1999a, Chapter 27).

This circuit has the recognizable elliptic topology that was invented and patented by Wilhelm Cauer in 1934, 1935, and 1936. The Cauer filter was a significant advance (both theoretically and commercially) over the earlier filter designs of Campbell, Zobel, Johnson, Butterworth, and Chebychev. For example, for one commercially important set of specifications for telephones, a fifth-order elliptic filter matches the behavior of a 17th-order Butterworth filter or an eighth-order Chebychev filter. The fifth-order elliptic filter has one less component than the eighth-order Chebychev filter. As Van Valkenburg (1982) relates in connection with the history of the elliptic filter:

> Cauer first used his new theory in solving a filter problem for the German telephone industry. His new design achieved specifications with one less inductor than had ever been done before. The world first learned of the Cauer method not through scholarly publication but through a patent disclosure, which eventually reached the Bell Laboratories. Legend has it that the entire Mathematics Department of Bell Laboratories spent the next two weeks at the New York Public Library studying elliptic functions. Cauer had studied


2
L27
103600uH
L40
155400uH
L31
155400uH
L45
155400uH
L47
155400uH
L56
51800uH
3
5
ZOUT1
V
VOUT1
RSOURCE
1K
RLOAD
1k
1
L34
51800uH
L50
51800uH
L43
51800uH
L53
51800uH
L14
51800uH
+
VSOURCE
−
C18
136nF
C12
136nF
C21
136nF
C3
136nF
C15
136nF
ZGND
0
4

10.3 FIGURE Evolved filter circuit with the Cauer (elliptic) topology.

mathematics under Hilbert at Goettingen, and so elliptic functions and their applications were familiar to him.

Genetic programming did not, of course, study mathematics under Hilbert or anybody else. Instead, the elliptic topology emerged from a run of genetic programming as a natural consequence of the problem's fitness measure and natural selection—not because the run was primed with domain knowledge about elliptic functions or filters or electrical circuitry. Genetic programming opportunistically *reinvented* the elliptic topology because necessity (fitness) is the mother of invention.

Genetic programming has also been used to evolve many other types of filters, including highpass, bandpass, bandstop, crossover, comb, and asymmetric filters (Koza et al. 1999a, 1999b).

10.4.4 Amplifier, Computational, Temperature-Sensing, Voltage Reference, and Other Circuits

Genetic programming has also been applied to the problem of automatic synthesis of both the topology and sizing of many analog electrical circuits composed of transistors. These include amplifiers (evolved using multiobjective fitness measures that consider gain, distortion, bandwidth, parts count, power consumption, and power supply rejection ratio), computational circuits (square root, squaring, cube root, cubing, logarithmic, and Gaussian), time-optimal controller circuits, source identification circuits, temperature-sensing circuits, and voltage reference circuits (Koza et al. 1999a, 1999b).

The amplifiers, computational circuits, electronic thermometers, and voltage reference circuits were all covered by one or more patents when they were first invented. Many of these circuits include previously patented subcircuits, such as Darlington emitter-follower sections (Darlington 1952).

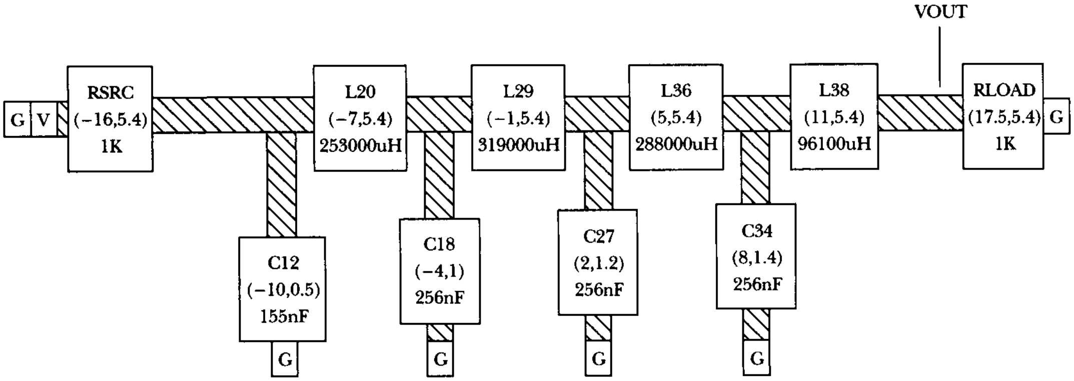
10.5 TOPOLOGY, SIZING, PLACEMENT, AND ROUTING OF CIRCUITS

Circuit *placement* involves the assignment of each of the circuit's components to a particular physical location on a printed circuit board or silicon wafer. *Routing* involves the assignment of a particular physical location to the wires between the leads of the circuit's components.

Genetic programming can simultaneously create a circuit's topology and sizing along with the placement and routing of all components as part of an integrated overall design process (Koza and Bennett 1999). It can do all four of these tasks while also optimizing additional considerations (such as minimizing the circuit's area).

This is accomplished by using an initial circuit that contains information about the geographic (physical) location of components and wires and using component-inserting and topology-modifying operations that appropriately adjust the geographic (physical) location of components and associated wires. For example, the initial circuit in the developmental process complies with the requirements that wires must not cross on a particular layer of a silicon chip or on a particular side of a printed circuit board, that there must be a wire connecting 100% of the leads of all the circuit's components, and that minimum clearance distances between wires, between components, and between wires and components must be maintained. Similarly, each of the circuit-constructing functions used preserves compliance with these requirements. Thus, every fully laid-out circuit complies with these requirements.

For example, in one run a lowpass filter circuit was first evolved in generation 25 for a discrete-component printed circuit board. The topology and



10.4 FIGURE Topology, sizing, placement, and routing of a lowpass filter for a printed circuit board.

component sizing for this circuit complied with all requirements (for passband ripple, stopband ripple, and attenuation); however, this circuit contained five capacitors and 11 inductors and occupied an area of 1775.2. Later, a 100%-compliant lowpass filter was created in generation 30 containing 10 inductors and five capacitors occupying an area of 950.3. Then, in generation 138, a physically compact lowpass filter circuit (Figure 10.4) containing four inductors and four capacitors and occupying an area of only 359.4 was created. As can be seen, this automatically created circuit has the Campbell topology (Campbell 1917).

10.6 AUTOMATIC SYNTHESIS OF CONTROLLERS BY MEANS OF GENETIC PROGRAMMING

The design of controllers is another area where there has been (prior to genetic programming) no previously known general technique for automatically creating the topology and tuning for a controller from a high-level statement of the design goals for the controller.

The purpose of a controller is to force, in a meritorious way, the actual response of a system (conventionally called the *plant*) to match a desired response (called the *reference signal*) (Astrom and Hagglund 1995; Boyd and Barratt 1991; Dorf and Bishop 1998).

In the PID type of controller, the controller's output is the sum of proportional (P), integrative (I), and derivative (D) terms based on the difference between the plant's output and the reference signal. The PID controller was patented in 1939 by Albert Callender and Allan Stevenson of Imperial Chemical Limited of Northwich, England.

Claim 1 covers what is now called the PI controller (Callender and Stevenson 1939):

> A system for the automatic control of a variable characteristic comprising means proportionally responsive to deviations of the characteristic from a desired value, compensating means for adjusting the value of the characteristic, and electrical means associated with and actuated by responsive variations in said responsive means, for operating the compensating means to correct such deviations in conformity with the sum of the extent of the deviation and the summation of the deviation.

Claim 3 covers what is now called the PID controller (Callender and Stevenson 1939):

> A system as set forth in claim 1 in which said operation is additionally controlled in conformity with the rate of such deviation.

The vast majority of automatic controllers used by industry are of the PID type. As Astrom and Hagglund (1995) observe: "Several studies . . . indicate the state of the art of industrial practice of control. The Japan Electric Measuring Instrument Manufacturing Association conducted a survey of the state of process control systems in 1989 . . . According to the survey, more than 90% of the control loops were of the PID type."

However, it is generally recognized by leading practitioners in the field of control that PID controllers are not ideal and that there are significant limitations on analytical techniques in designing controllers. As Boyd and Barratt (1991) stated in *Linear Controller Design: Limits of Performance:* "The challenge for controller design is to productively use the enormous computing power available. Many current methods of computer-aided controller design simply automate procedures developed in the 1930's through the 1950's . . ."

There is no preexisting general-purpose analytic method (prior to genetic programming) for automatically creating a controller for arbitrary linear and nonlinear plants that can simultaneously optimize prespecified performance metrics (such as minimizing the time required to bring the plant output to the desired value as measured by, say, the integral of the time-weighted absolute error), satisfy time-domain constraints (involving, say, overshoot and disturbance rejection), satisfy frequency domain constraints (e.g., bandwidth), and satisfy additional constraints, such as constraints on the magnitude of the control variable and the plant's internal state variables.

10.6.1 Robust Controller for a Two-Lag Plant

We employ a problem involving control of a two-lag plant to illustrate the automatic synthesis of controllers by means of genetic programming (Koza et al. 2000). The problem here (described by Dorf and Bishop 1998, p. 707) is to create both the topology and parameter values for a controller for a two-lag plant such that plant output reaches the level of the reference signal so as to minimize the integral of the time-weighted absolute error (ITAE), such that the overshoot in response to a step input is less than 2%, and such that the controller is robust in the face of significant variation in the plant's internal gain, K, and the plant's time constant, τ.

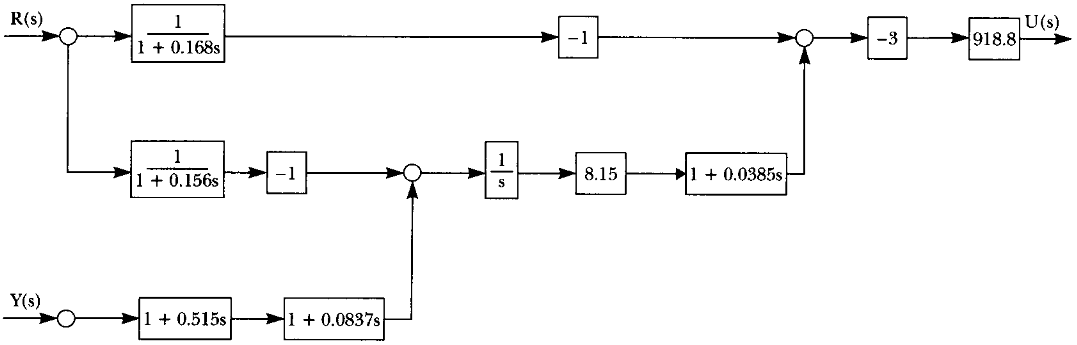
Genetic programming routinely creates PI and PID controllers infringing on the 1942 Callender and Stevenson patent during intermediate generations of runs of genetic programming on controller problems. However, the PID controller is not the best controller for this problem (and many others).

Figure 10.5 shows the block diagram for the best-of-run controller evolved on generation 32 of one run of the two-lag plant problem. In this figure, $R(s)$ is the reference signal; $Y(s)$ is the plant output; and $U(s)$ is the controller's output (control variable).

The controller evolved by genetic programming differs from a conventional PID controller in that the genetically evolved controller employs a second derivative processing block. As will be seen, this evolved controller is 2.42 times better than the Dorf and Bishop (1998) controller as measured by the criterion used by Dorf and Bishop (namely, the integral of the time-weighted absolute error). In addition, this evolved controller has only 56% of the rise time in response to the reference input, has only 32% of the settling time, and is 8.97 times better in terms of suppressing the effects of disturbance at the plant input.

After applying standard manipulations to the block diagram of this evolved controller, the transfer function for the best-of-run controller from generation 32 for the two-lag plant can be expressed as a transfer function for a prefilter and a transfer function for a compensator. The transfer function for the prefilter, $G_{p32}(s)$, for the best-of-run individual from generation 32 is

$$G_{p32}(s)=\frac{1(1+.1262s)(1+.2029s)}{(1+.03851s)(1+.05146)(1+.08375)(1+.1561s)(1+.1680s)}$$



10.5 FIGURE Best-of-run genetically evolved controller.

The transfer function for the compensator, $G_{c32}(s)$, is

$$G_{c32}(s) = \frac{7487(1+.03851s)(1+.05146s)(1+.08375s)}{s}$$
$$= \frac{7487.05+1300.63s+71.2511s^2+1.2426s^3}{s}$$

The s^3 term (in conjunction with the s in the denominator) indicates a second derivative. Thus, the compensator consists of a second derivative in addition to proportional, integrative, and derivative functions. Harry Jones of the Brown Instrument Company of Philadelphia patented this same kind of controller topology in 1942.

Claim 38 of the Jones 1942 patent states:

> In a control system, an electrical network, means to adjust said network in response to changes in a variable condition to be controlled, control means responsive to network adjustments to control said condition, reset means including a reactance in said network adapted following an adjustment of said network by said first means to initiate an additional network adjustment in the same sense, and rate control means included in said network adapted to control the effect of the first mentioned adjustment in accordance with the second or higher derivative of the magnitude of the condition with respect to time.

Note that the user of genetic programming did not preordain, prior to the run (as part of the preparatory steps for genetic programming), that a second derivative should be used in the controller (or, for that matter, that a P, I, or D block should be used). The evolutionary process discovered that these elements were helpful in producing a good controller for this problem. That is, necessity was the mother of invention. Similarly, the user did not preordain any particular topological arrangement of proportional, integrative, derivative, second derivative, or other functions within the automatically created controller. Instead, genetic programming automatically created a robust controller for the given plant without the benefit of user-supplied information concerning the total number of processing blocks to be employed in the controller, the type of each processing block, the topological interconnections between the blocks, the values of parameters for the blocks, or the existence of internal feedback (none in this instance) within the controller.

10.7 THE ILLOGICAL NATURE OF CREATIVITY AND EVOLUTION

Many computer scientists and mathematicians unquestioningly assume that every problem-solving technique must be logically sound, deterministic, logically consistent, and parsimonious. Accordingly, most conventional methods of artificial intelligence and machine learning are constructed so as to possess these characteristics.

However, logic does not govern two of the most important and significant types of processes for solving complex problems: the invention process (performed by creative humans) and the evolutionary process (occurring in nature).

The biological metaphor underlying genetic programming is very different from the underpinnings of all other techniques that have previously been tried in pursuit of the goal of automatically creating computer programs.

A new idea that can be logically deduced from facts that are known in a field, using transformations that are known in a field, is not considered to be an invention. There must be what the patent law refers to as an "illogical step" (i.e., an unjustified step) to distinguish a putative invention from that which is readily deducible from that which is already known. Humans supply the critical ingredient of "illogic" to the invention process. Interestingly, everyday usage parallels the patent law concerning inventiveness: People who mechanically apply existing facts in well-known ways are summarily dismissed as being uncreative. Logical thinking is unquestionably useful for many purposes. It usually plays an important role in setting the stage for an invention. But, at the end of the day, logical thinking is not sufficient in the invention process.

Recalling his invention in 1927 of the negative feedback amplifier, Harold S. Black said:

> Then came the morning of Tuesday, August 2, 1927, when the concept of the negative feedback amplifier came to me in a flash while I was crossing the Hudson River on the Lackawanna Ferry, on my way to work. For more than 50 years, I have pondered how and why the idea came, and I can't say any more today than I could that morning. All I know is that after several years of hard work on the problem, I suddenly realized that if I fed the amplifier output back to the input, in reverse phase, and kept the device from oscillating (singing, as we called it then), I would have exactly what I wanted: a means of canceling out the distortion of the output. I opened my morning newspaper and on a page of *The New York Times* I sketched a simple canonical diagram of a negative feedback amplifier plus the equations for the amplification with feedback (Black 1977).

Of course, inventors are not oblivious to logic and knowledge. They do not thrash around using blind random search. Black did not try to construct the negative feedback amplifier from neon bulbs or doorbells. Instead, "several years of hard work on the problem" set the stage and brought his thinking into the proximity of a solution. Then, at the critical moment, Black made his "illogical" leap. This unjustified leap constituted the invention.

The design of complex entities by the evolutionary process in nature is another important type of problem solving that is not governed by logic. In nature, solutions to design problems are discovered by the probabilistic process of evolution and natural selection. This process is not guided by mathematical logic. Indeed, inconsistent and contradictory alternatives abound. In fact, such genetic diversity is necessary for the evolutionary process to succeed. Significantly, the solutions evolved by evolution and natural selection almost always differ from those created by conventional methods of artificial intelligence and machine learning in one very important respect. Evolved solutions are not brittle; they are usually able to grapple with the perpetual novelty of real environments.

Since genetic programming is a probabilistic process that is not encumbered by the preconceptions that often channel human thinking down familiar paths, it often creates novel designs.

Similarly, genetic programming is not guided by the inference methods of formal logic in its search for a computer program to solve a given problem. When the goal is the automatic creation of computer programs, we believe that the nonlogical approach used in the invention process and in natural evolution is far more fruitful than are the logic-driven and knowledge-based principles of conventional artificial intelligence and machine learning. In short, "logic considered harmful."

10.8 CONCLUSIONS

This chapter has demonstrated that a technique of evolutionary computation (genetic programming) can be successfully used to automatically create computer programs and designs from a high-level statement of a problem's requirements. The results in this chapter (and the other recently produced human-competitive results in Table 10.1) demonstrate that genetic programming can produce results that are competitive with human-produced results in a variety of areas. Both the number of these results and the variety of areas further suggest that genetic programming is on the threshold of routinely producing human-competitive results. We expect that the rapidly decreasing cost of computing power will enable genetic programming to deliver additional human-competitive results on increasingly difficult problems and, in particular, that genetic programming will be routinely used as an "invention machine" for producing patentable new inventions.

REFERENCES

Aaserud, O., and I. Nielsen (1995). Trends in current analog design: A panel debate. *Analog Integrated Circuits and Signal Processing* 7(1):5–9.

Andre, D., F. H. Bennett III, and J. R. Koza (1996). Discovery by Genetic Programming of a Cellular Automata Rule That Is Better Than Any Known Rule for the Majority Classification Problem. In J. R. Koza, D. E. Goldberg, D. B. Fogel, and R. L. Riolo (eds.), *Genetic Programming 1996: Proceedings of the First Annual Conference,* July 28–31, 1996, Stanford University, MIT Press, pp. 3–11.

Andre, D., and A. Teller (1998). Evolving Team Darwin United. In Asada, Minoru (ed), *RoboCup-98: Robot Soccer World Cup II.* Lecture Notes in Computer Science, Springer-Verlag.

Astrom, K. J., and T. Hagglund (1995). *PID Controllers: Theory, Design, and Tuning.* Second edition. Instrument Society of America.

Banzhaf, W., J. Daida, and A. E. Eiben (eds.) (1999). *Proceedings of the Genetic and Evolutionary Computation Conference.* Morgan Kaufmann.

Banzhaf, W., P. Nordin, R. E. Keller, and F. D. Francone (1998). *Genetic Programming—An Introduction.* Morgan Kaufmann.

Black, H. S. (1977). Inventing the Negative Feedback Amplifier. *IEEE Spectrum* (Dec.):55–60.

Boyd, S. P., and C. H. Barratt (1991). *Linear Controller Design: Limits of Performance.* Prentice-Hall.

Callender, A., and A. B. Stevenson (1939). *Automatic Control of Variable Physical Characteristics.* U.S. Patent 2,175,985. Filed February 17, 1936, in United States. Filed February 13, 1935, in Great Britain. Issued October 10, 1939, in United States.

Campbell, G. A. (1917). *Electric Wave Filter.* U.S. Patent 1,227,113. Filed July 15, 1915. Issued May 22, 1917.

Cauer, W. (1934). *Artificial Network.* U.S. Patent 1,958,742. Filed June 8, 1928, in Germany. Filed December 1, 1930, in United States. Issued May 15, 1934, in United States.

Cauer, W. (1935). *Electric Wave Filter.* U.S. Patent 1,989,545. Filed June 8, 1928, in Germany. Filed December 6, 1930, in United States. Issued January 29, 1935, in United States.

Cauer, W. (1936). *Unsymmetrical Electric Wave Filter.* Filed November 10, 1932, in Germany. Filed November 23, 1933, in United States. Issued July 21, 1936, in United States.

Darlington, S. (1952). *Semiconductor Signal Translating Device.* U.S. Patent 2,663,806. Filed May 9, 1952. Issued December 22, 1953.

Dorf, R. C., and R. H. Bishop (1998). *Modern Control Systems.* Eighth edition. Addison-Wesley.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution.* John Wiley & Sons.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Ince, D. C. (ed.) (1992). *Mechanical Intelligence: Collected Works of A. M. Turing.* North Holland.

Jones, H. S. (1942). *Control Apparatus.* U.S. Patent 2,282,726. Filed October 25, 1939. Issued May 12, 1942.

Koza, J. R. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Koza, J. R. (1994a). *Genetic Programming II: Automatic Discovery of Reusable Programs.* MIT Press.

Koza, J. R. (1994b). *Genetic Programming II Videotape: The Next Generation.* MIT Press.

Koza, J. R., W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (eds.) (1998). *Genetic Programming 1998: Proceedings of the Third Annual Conference.* Morgan Kaufmann.

Koza, J. R., and F. H. Bennett III (1999). Automatic Synthesis, Placement, and Routing of Electrical Circuits by Means of Genetic Programming. In L. Spector, W. B. Langdon, U.-M. O'Reilly, and P. Angeline (eds.), *Advances in Genetic Programming 3,* MIT Press, Chapter 6, pp. 105–134.

Koza, J. R., F. H. Bennett III, D. Andre, and M. A. Keane (1999a). *Genetic Programming III: Darwinian Invention and Problem Solving.* Morgan Kaufmann.

Koza, J. R., F. H. Bennett III, D. Andre, M. A. Keane, and S. Brave (1999b). *Genetic Programming III Videotape.* Morgan Kaufmann.

Koza, J. R., M. A. Keane, J. Yu, F. H. Bennett III, and W. Mydlowec (2000). Automatic Creation of Human-Competitive Programs and Controllers by Means of Genetic Programming. *Genetic Programming and Evolvable Machines* 1(1–2):121–164.

Koza, J. R., and J. P. Rice (1992). *Genetic Programming: The Movie.* MIT Press.

Langdon, W. B. (1998). *Genetic Programming and Data Structures: Genetic Programming + Data Structures = Automatic Programming!* Kluwer.

O'Connor, D. G., and R. J. Nelson (1962). *Sorting System with N-Line Sorting Switch.* U.S. Patent 3,029,413. Issued April 10, 1962.

Poli, R., P. Nordin, W. B. Langdon, and T. C. Fogarty (1999). *Genetic Programming: Second European Workshop. EuroGP'99.* Goteborg, Sweden, May 1999. Lecture Notes in Computer Science 1598. Springer-Verlag.

Samuel, A. L. (1959). Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 3(3):210–229.

Samuel, A. L. (1983). AI: Where It Has Been and Where It Is Going. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence,* Morgan Kaufmann, pp. 1152–1157.

Spector, L., H. Barnum, and H. J. Bernstein (1998). Genetic Programming for Quantum Computers. In J. R. Koza, W. Banzhaf, K. Chellapilla, K. Deb, M. Dorigo, D. B. Fogel, M. H. Garzon, D. E. Goldberg, H. Iba, and R. Riolo (eds.), *Genetic Programming 1998: Proceedings of the Third Annual Conference,* Morgan Kaufmann, pp. 365–373.

Spector, L., H. Barnum, and H. J. Bernstein (1999). Quantum Computing Applications of Genetic Programming. In L. Spector, W. R. Langdon, U.-M. O'Reilly and P. Angeline (eds.), *Advances in Genetic Programming 3,* MIT Press, pp. 135–160.

Spector, L., H. Barnum, H. J. Bernstein, and N. Swamy (1999). Finding a Better-Than-Classical Quantum AND/OR Algorithm Using Genetic Programming. In *Proceedings of 1999 Congress on Evolutionary Computation,* IEEE Press, pp. 2239–2246.

Spector, L., W. B. Langdon, U.-M. O'Reilly, and P. Angeline (eds.) (1999). *Advances in Genetic Programming 3.* MIT Press.

Van Valkenburg, M. E. (1982). *Analog Filter Design.* Harcourt Brace Jovanovich.

Zobel, O. J. (1925). *Wave Filter.* U.S. Patent 1,538,964. Filed January 15, 1921. Issued May 26, 1925.

11 CHAPTER

Toward a Symbiotic Coevolutionary Approach to Architecture

Helen Jackson Coevolution

11.1 INTRODUCTION

This chapter builds on earlier work using genetic programming (GP) and a Lindenmayer system (L-system) representation within the sphere of generative architectural design. L-systems are explained briefly and two contrasting embryology strategies are outlined. Artificial selection is discussed, and the wide divergence of opinion as to what might constitute an architectural configuration illustrated. Examples of successful single-goal evolution are presented, with the space syntax measure of *integration* investigated as a generic identifier of architectural form. Dual- and multigoal evolution are considered within the context of the architectural design discipline. It is suggested that an appropriate response to the complex nature of architectural organisms is the development of a symbiotic coevolutionary metaphor where interwoven systems within architecture are viewed as mutual species. The classification of these species leads toward a more architecture-specific genetic code. An outline of future work intended to develop such a representation begins with the identification of a naïve architectural form representation and summarizes a gradual process for the refinement of this representation into a genuinely useful encoding of architectural form.

11.2 LINDENMAYER SYSTEMS

Lindenmayer systems, introduced as a method for modeling plant development (Lindenmayer 1968), are capable of providing a compressed description of any continuous self-similar form. An L-system consists of three elements: an *alphabet* of "words" allowed in the system, an initial *axiom*, and a *set of productions*. "Words" are strings consisting of a finite number of symbols in combination; both axiom and productions must be viable words. The productions are *rewrite rules*—at each rewrite all copies of a particular symbol are replaced by the entire production. It takes very few iterations to produce a genotype of considerable complexity. The L-system alphabet used in these experiments comprises symbols based on a system of turtle geometry, which allows L-system words (the genotype) to be interpreted as two- or three-dimensional forms (the phenotype). The symbols making up the alphabet are LISP functions, so productions are LISP S-expressions. This allows genetic operations to be carried out on the L-system programs using the basic genetic programming operators of recombination and mutation (Koza 1992).

11.2.1 Example L-Systems

Using the system outlined above, it is possible to depict a large range of fractal curves. For example, changes in the initial axiom, production, and rotation angle of the Koch curve produce the variations shown in Figure 11.1 (Prusinkiewicz and Lindenmayer 1990; Jackson 1998).

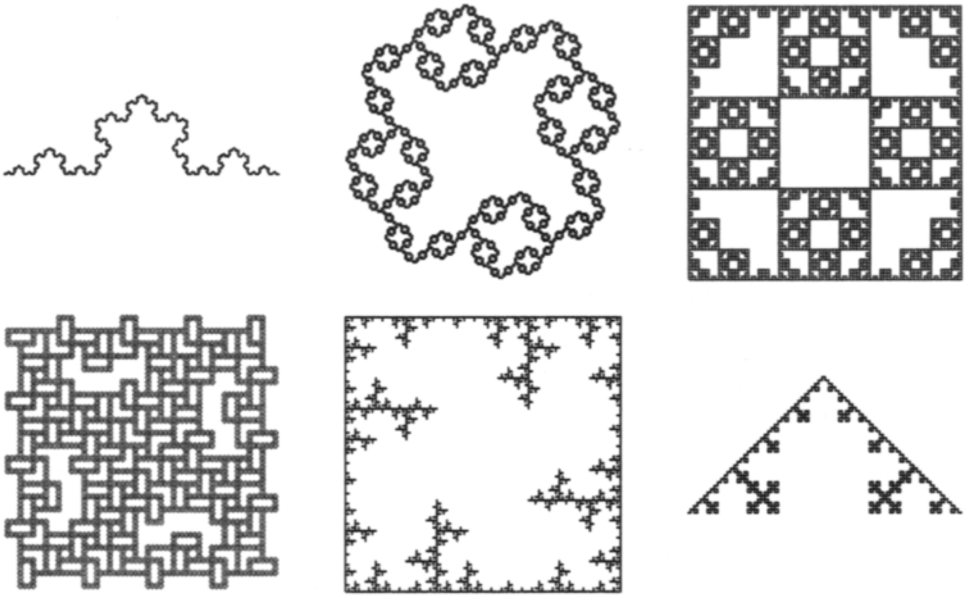
The same methods can be used to produce edge-rewriting L-systems with more than one production rule. To illustrate, the Sierpiński gasket is expanded below and then interpreted on a two-dimensional isospatial grid (Figure 11.2):

Axiom: F_r
Productions: F_l → F_r PLUS F_l PLUS F_r
F_r → F_l MINUS F_r MINUS F_l

F_r →
F_l MINUS F_r MINUS F_l →
F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r →

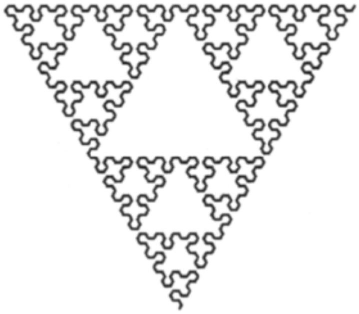
F_l MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l →

F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l



11.1
FIGURE

Classic L-systems produced using the turtle interpretation: Koch curves with varying productions.



11.2
FIGURE

The Sierpiñski gasket (Prusinkiewicz and Lindenmayer 1990; Jackson 1998).

PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l
PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l PLUS F_r PLUS F_l
MINUS F_r MINUS F_l PLUS F_r PLUS F_l PLUS F_r PLUS F_l MINUS F_r MINUS F_l
PLUS F_r PLUS F_l PLUS F_r MINUS F_l MINUS F_r MINUS F_l MINUS F_r PLUS F_l
PLUS F_r

11.2.2 The Isospatial Grid

These early GP/L-system experiments interpret the phenotypes within a two- or three-dimensional isospatial grid (see Figure 11.3), rather than the Cartesian grid traditional in CAD packages (Broughton, Coates, and Jackson 1998). The isospatial grid consists of a point and its 12 equidistant neighbors, thus removing the predisposition against diagonal relationships found in the Cartesian grid (Frazer 1995). Right-angle relationships in three-dimensions are possible but are not presumed. The intention is to avoid generative form that is entirely an artifact of the representation.

11.2.3 Spatial Embryology

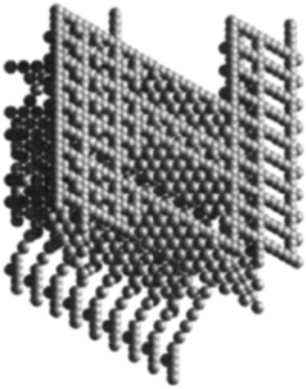
The mapping from genotype to phenotype plays a critical part in the success of an encoding. The simple L-system genotype introduced as a neutral form generator can also be used to produce a significant number of alternative phenotypic outcomes. The advantage of adding complexity at this level is that the genotype remains an extremely condensed form descriptor. A number of different L-system embryologies have been tested, giving a wide range of phenotypes.

An embryology that results in orthogonal spatial configurations rather than isospatial forms is currently under investigation. An edge-rewriting L-system is used, with an alphabet of six symbols, shown in Figure 11.4. Each genotype consists of a pair of production rules.

The rectangular forms are interpreted as plan representations of spaces: F_l produces linear movement spaces, and F_r convex occupation spaces (Hillier 1996). Rectangles that are directly adjacent or overlapping can combine to produce large spaces, according to a rule table, allowing variability in room sizes. For example, the genotype (production rules) and basic form configuration shown in Figure 11.5 would be interpreted as the phenotype in Figure 11.6.

11.3 ARTIFICIAL SELECTION

Artificial selection allows the creativity and skills of the architect to direct the evolutionary path. Humans can make decisions about whether a space is attractive and pleasant, or chaotic and claustrophobic, almost instantaneously. These decisions are based on an underlying scheme of value judgments that are not



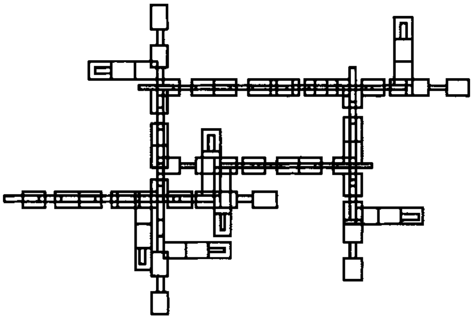
11.3 FIGURE Example of three-dimensional isospatial L-system phenotype.

F_l	Move forward a step and insert a rectangle of height:width ratio 1:3.	▭
F_r	Move forward and insert a rectangle of height:width ratio 3:4.	□
PLUS	Rotate heading counterclockwise by 90° relative to current heading.	
MINUS	Rotate heading clockwise by 90° relative to current heading.	
( )	Brackets indicate branching points, allowing tree structures to be described.	

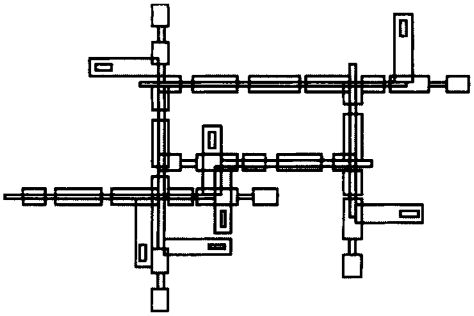
11.4 FIGURE Edge-rewriting L-system alphabet.

usually made explicit. It is proposed that these implicit judgments are interrogated through the medium of artificial selection; that is, an evolutionary process where the only fitness function is the "eyeball test"—a human choosing preferred forms from a breeding pool to produce offspring. The way in which forms are chosen is under investigation—what makes one form more "architectural" than another? Experimental results are proposed as a possible means of developing a set of rules formulated from human responses. Such rules form the basis of a discussion on how people perceive and understand architecture. An evaluation of successful genotypes could also lead to the identification of genetic structures

$F_r \rightarrow$ (F_l F_l (F_l (F_r F_l F_r MINUS PLUS) PLUS F_r F_r (F_r MINUS PLUS) F_l) PLUS)
$F_l \rightarrow$ (PLUS (F_l F_l F_r) F_r F_l F_l (F_l PLUS PLUS) MINUS)



11.5 FIGURE Example L-system genotype and basic form configuration.

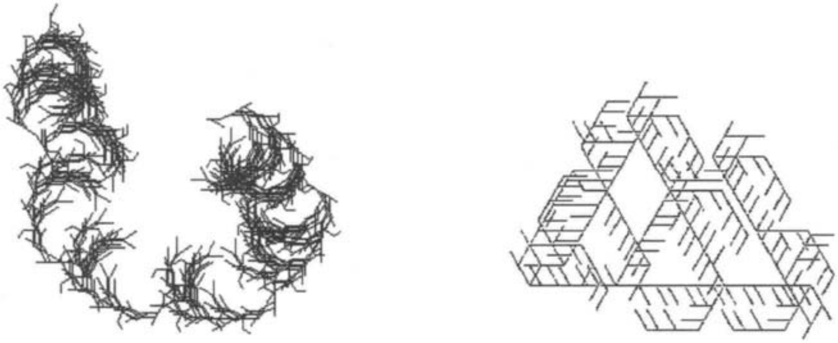


11.6 FIGURE Interpreted phenotype.

common to many architecturally preferred solutions. A second level of genetic code made up of these building blocks could be developed, as in the genetic engineering paradigm (Gero 1998).

11.3.1 The Eyeball Test

To investigate this, two human subjects were asked to direct the course of an evolutionary run. Both users were presented with the same initial pool, containing 10 two-dimensional line drawings, and were asked to select the two most



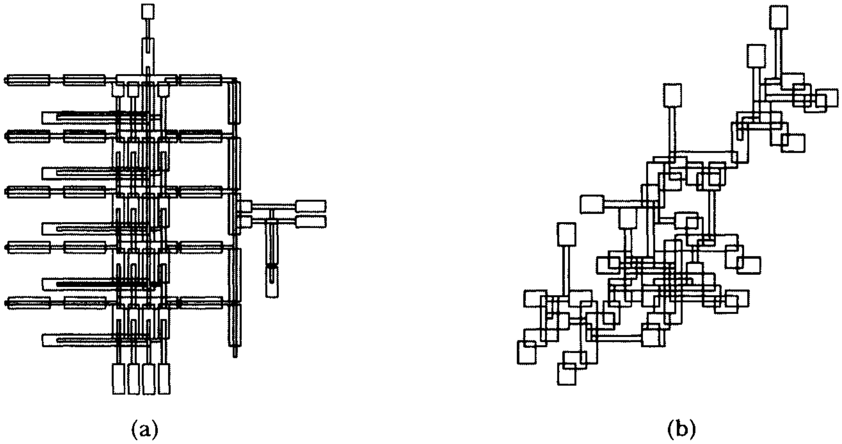
11.7 FIGURE Best-of-run individuals evolved over 10 generations from the same initial pool by two subjects using the "eyeball test."

"architectural" plan configurations after seeing each member of the pool for only five seconds. A new generation of individuals was produced through crossover and mutation of the chosen parents and the process repeated. After 10 generations, a single favored outcome was chosen by the subject. The two resulting configurations are shown in Figure 11.7.

It can be seen that the forms chosen in this experiment differ widely. However, each user had no difficulty in choosing configurations at each stage, despite the short time period allowed for viewing individuals. There are implicit criteria underlying this process of decision making. Natural selection within the virtual world of the computer can only take place once some or all of these have been identified.

11.4 SINGLE-GOAL EVOLUTION

Phenotypes produced through the orthogonal spatial interpretation of L-system genotypes provide a way of evaluating some of the criteria upon which artificial selection is based. Two randomly produced phenotypes are shown in Figure 11.8; at a glance one decides that Figure 11.8(a) is potentially quite architectural, whereas Figure 11.8(b) is not, although the two are made up of combinations of the same building blocks.



11.8 FIGURE Two randomly produced orthogonal spatial phenotypes.

11.4.1 "Generic Function" as Fitness Function

Hillier (1996) has shown that integration[1] acts as a generic function for architecture. The majority of buildings have a highly integrating configuration; that is, they have a low "integration" value or i-value,[2] whatever their function:

> Analysis of large numbers of buildings over a number of years suggests that around 150 cells [rooms], virtually all buildings will be within the shallowest 20 per cent of the range of possibility [the most integrating end of the range], and most much below it, at 300 cells, nearly all will be within the bottom 10 per cent, and at around 500 most will be within the bottom 5 per cent (Hillier 1996, p. 283).

It seems possible that this "generic function" could be a part of the unconscious criteria by which one makes architectural judgements on spatial configurations. This is either through intuitive recognition of well-integrated systems or comparison with familiar highly integrated spatial layouts. Figure 11.8(b) has an i-value approximately 3.5 times that of Figure 11.8(a) (a low i-value indicates a highly integrating configuration). Figure 11.8(a) is therefore the more integrated and more "architectural" configuration using this system.

---

[1] ". . . the topological depth of each space from all others . . ." (Hillier 1996, p. 283).

[2] Relative asymmetry (integration value) $= \frac{2(MD-1)}{k-2}$

where MD is the mean depth and k the number of spaces in the system (Hillier and Hanson 1984, p. 108).

11.4.2 Evolution toward Low i-Values

Figure 11.9 shows the results of an evolutionary run where the fitness function was the minimization of i-values. A range of different solutions have been produced. None of the solutions are immediately nonarchitectural, unlike the randomly produced phenotype in Figure 11.8(b), so the use of Hillier's generic function as a fitness criteria has provided a way of selecting for potentially architectural configurations.

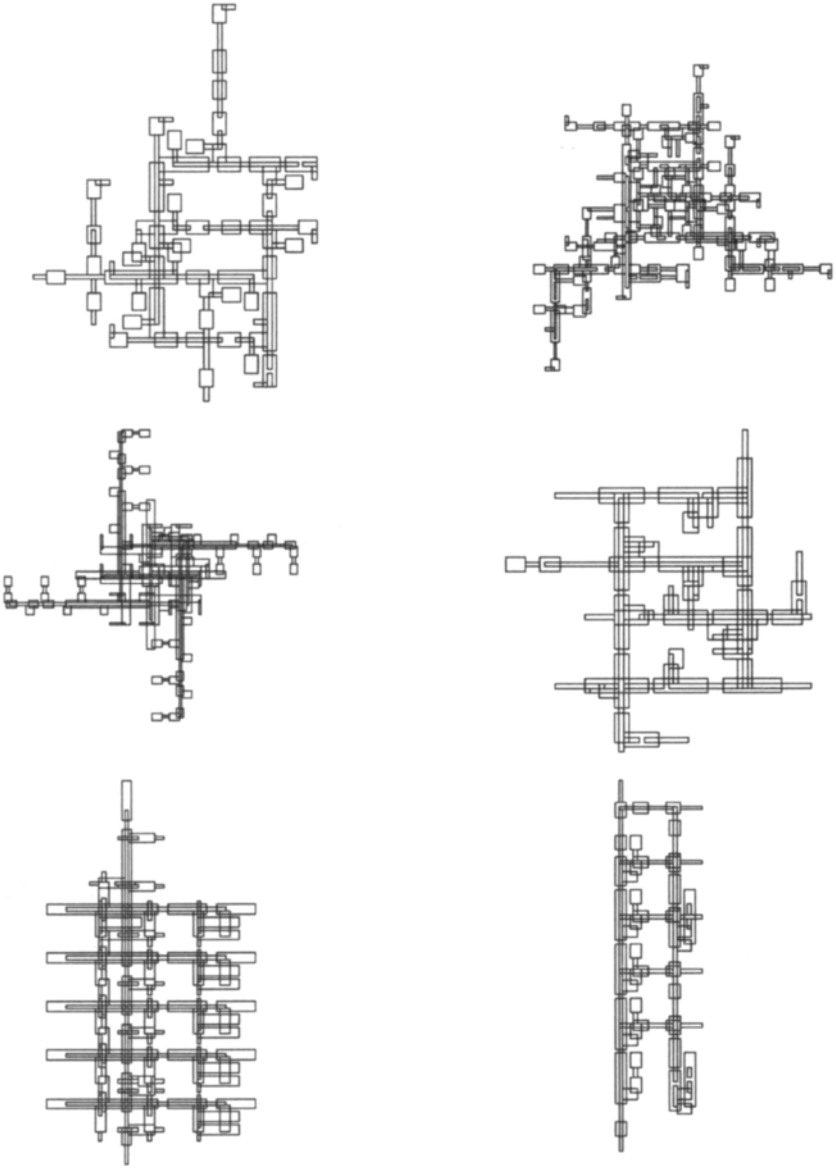
11.4.3 Structural Stability

Successful evolution using an L-system GP with three-dimensional isospatial phenotypes has been demonstrated using a fitness criterion based on structural stability (Jackson 1998). The structural rules used are an idealization of real-world situations, not taking material properties into consideration, intended to test the generative process rather than to engineer physical form.

Three-dimensional phenotypes were tested using three criteria relating to structural stability and vertical loading. Figure 11.10 shows that components such as buttresses, columns, beams, and cantilevers are generated, producing conceptual potential structures.

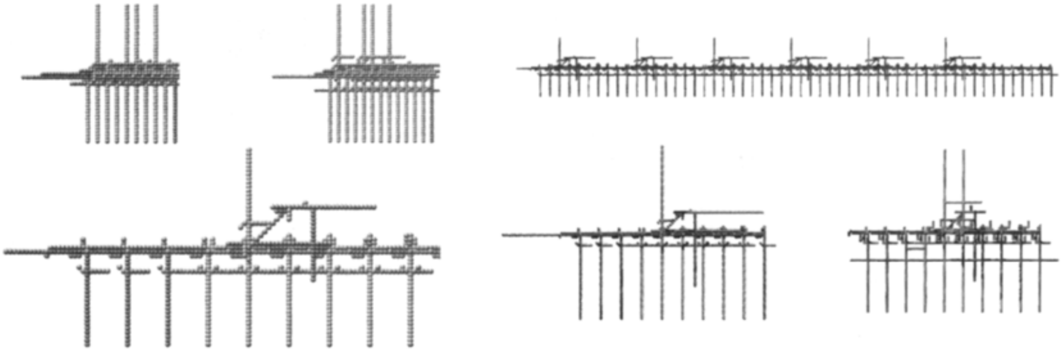
11.4.4 Architecture As a Multigoal Task

In addition to the space syntax and structural criteria described above, other single-goal fitness functions used with the GP/L-system have been shown to produce useful forms. Achievements include the successful capture of virtual "flies" (Broughton, Coates, and Tan 1997) and the enclosure of space (Broughton, Coates, and Jackson 1998). However, the domain of architecture is not single criterion. The best architects design according to numerous measures, most of them implicit rather than explicitly stated, ranging from the mundane to the esoteric.



11.9
FIGURE

Evolved spatial configurations with low integration values.



11.10 FIGURE Family of evolved three-dimensional isospatial structures in elevation (Jackson 1998).

11.4.5 Dual-Goal Evolution

A prototype system has been developed that defines useful forms as those that both enclose a number of spaces and allow (virtual) people access to those spaces (Jackson 1998). The outcomes of trial two-dimensional evolutionary runs were layouts comprising connected series of spaces, demonstrating that dual-goal evolution could work with the GP/L-system method.

Further experiments using more than two fitness criteria showed a markedly lower success rate, seemingly because of the number and disparity of criteria required. Within the discipline of architecture it is common for two design pressures to act in opposition to one another. This polarity and the large space of potential criteria make multigoal scenarios extremely complicated.

11.5 REPRESENTATION, SYSTEMS, AND SYMBIOSIS

GP using a neutral form generator such as the isospatial L-system locates all architectural judgment in the fitness functions. While all potential works of architecture could theoretically be described using this representation, numerous nonarchitectural forms can also be represented. The search space is therefore significantly wider than the space of possible architecture. A representation capable of describing all possible pieces of architecture and only possible pieces of architecture would reduce the search space, cutting evolutionary time and processing power. However, a poor representation will constrict the search space overly, eliminating the possibility of novel results.

The development of a genetic architectural representation has been a focus for investigation for some years. Soddu (1994) describes an evolutionary system where the individuals are members of a species that is a small subset of the architectural search space. This approach resembles an evolutionary shape grammar and allows detailed exploration of a single design concept through a large number of virtually realized designs. (More details can be found in Chapter 2 in this book.) Rosenman (1997) discusses the need for a design grammar to be used within a genetic approach and presents an orthogonal two-dimensional grammar for the evolution of house designs.

11.5.1 Coevolution

An alternative to the multigoal method in which large numbers of design pressures are stated and combined can be found in a coevolutionary approach. This requires the identification of separate, but interlinked, systems within architecture. These can be treated as individual species, locked in a symbiotic partnership. Each can only evolve to increase its fitness through coadaptation with others. Individual systems have a small number of applicable design criteria. The complexity of the model depends not on the interaction of fitness scores, but on the interspecies interaction.

It is proposed that a gradual process of prototyping could allow the development of a more architecture-specific genetic code, made up of representations for a number of interwoven systemic species. A naïve architectural representation can be described and its evolutionary performance tested at all stages. Testing is through comparison with a neutral form generator, allowing development of more sophisticated representations. The initial experimental outline, aiming to provide proof of concept, is two-dimensional for clarity and to reduce processing power requirements. Both artificial and natural selection play a part in this process.

11.5.2 Naïve Architectural Form Representation

A statement of three interlinked systems and levels is proposed as a starting point for this prototype. The nature of the process is such that this representation will undergo gradual refinement.

The lowest level, with low genetic diversity, concerns spatial properties. Space itself is evolved and the potential forms must meet fundamental criteria such as configurational properties (i-value) and accessibility for humans plus project-specific criteria such as floor area or site requirements. There is some genetic diversity in the breeding pool, but the search space is usefully constrained.

A middle level concerns the movement system. The circulatory route of a building is required to link spaces in a way that enables their configurational requirements. However, there is a wide range of possibilities in terms of the forms capable of creating particular configurations. There is more genetic diversity than in the spaces themselves. All fully formed circulatory systems will meet the defined requirements.

The top level, with the maximum genetic diversity, is concerned with the physical and visual envelope of the form. This can have many manifestations while still meeting the underlying definition of "architecture." This level is concerned with more personal notions of style, visual impact, and image and may need to be evaluated through direct interaction with a human designer.

11.5.3 Spatial Embryology

The orthogonal spatial embryology is an early experiment introducing a more architecture-specific representation. Following Hillier (1996), space is divided into occupation and movement, with a basic relationship of spatial type to proportion. The simplified, orthogonal nature of this particular representation means that it is, of course, far too restrictive to be able to describe adequately the wide range of architectural forms. However, for the purposes of testing ideas about architectural representation, this embryology provides a starting point. Its advantage is that all the forms produced have some architectural nature—they all describe an enclosed system containing movement and occupation spaces—while the isospatial L-system forms, for example, may not even enclose space.

The two spatial types involved in this embryology correspond to the lowest two levels of the architectural representation outlined above. The evolutionary run illustrated in Figure 11.9 uses only the i-value criterion. If the two spatial types are divided into separate species, an additional criterion can be used within evolution without the need for optimization techniques. Performance of the dual-species model can be compared with that of the single-species dual-goal approach.

11.6 CONCLUSIONS

It has been seen that an evolutionary process is applicable to the bottom-up generative design of architectural form. The GP/L-system method has been shown to produce useful global outcomes when a single fitness function is used. Two different L-system embryologies have been considered and a range of fitness functions discussed, from the "eyeball test" to structural rules. Integration, a property of spatial arrangements, is presented as a generic fitness function for architectural configurations of space. The problems encountered when attempting to describe architecture or map explicitly its criteria are discussed. A model is proposed for the description of architectural forms as a set of symbiotic relationships. The intention is to test this model through experimentation and comparison with other evolutionary approaches and representations.

ACKNOWLEDGMENTS

The author would like to thank Richard Coyne, John Lee, and Eileen Jackson for their useful comments on an earlier version of this chapter.

REFERENCES

Broughton, T., P. Coates, and H. Jackson (1998). Evolutionary Models of Space. In *Eurographics UK 1998, Conference Proceedings,* University of Leeds.

Broughton, T., P. Coates, and A. Tan (1997). The Use of Genetic Programming in Exploring 3D Design Worlds. In R. Junge (ed.), *CAAD Futures 1997,* Kluwer Academic Publishers.

Dawkins, R. (1996). *Climbing Mount Improbable.* Viking.

Frazer, J. (1995), *An Evolutionary Architecture.* Architectural Association, London.

Gero, J. (1998). Adaptive Systems in Designing: New Analogies from Genetics and Developmental Biology. In I. Parmee (ed.), *Adaptive Computing in Design and Manufacture,* Springer, pp. 3–12.

Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Hillier, B. (1996). *Space Is the Machine: A Configurational Theory of Architecture.* Cambridge University Press.

Hillier, B., and J. Hanson. (1984). *The Social Logic of Space.* Cambridge University Press.

Jackson, H. (1998). An Evolutionary Step. MSc thesis (unpublished), University of East London.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Lindenmayer, A. (1968). Mathematical Models for Cellular Interaction in Development, Parts I and II. *Journal of Theoretical Biology* 18:280–315.

Prusinkiewicz, P., and A. Lindenmayer. (1990). *The Algorithmic Beauty of Plants.* Springer-Verlag.

Rosenman, M. A. (1997). The Generation of Form Using an Evolutionary Approach. In D. Dasgupta and Z. Michalewicz (eds.), *Evolutionary Algorithms in Engineering Applications,* Springer-Verlag, pp. 69–85. Downloaded from *http://www.arch.su.edu.au/~mike/publications.html,* February 1999.

Soddu, C. (1994). The Design of Morphogenesis: An Experimental Research about the Logical Procedures in Design Processes. *DEMETRA Magazine* #1. Downloaded from *http://soddu2.dst. polimi.it/demetra2.htm,* February 1999.

This Page Intentionally Left Blank

12

CHAPTER

Using Evolutionary Algorithms to Aid Designers of Architectural Structures

Peter von Buelow University of Stuttgart

12.1 INTRODUCTION

In engineering fields, accomplishing an objective with a minimum of effort, either in terms of material, time, or other expense, is a basic activity. For this reason it is easy to understand the interest designers have in different optimization techniques. Mathematical as well as model-based tools have traditionally been employed for such optimization. In recent times, mathematical methods executed on computers have become predominant. Unfortunately, computer-derived solutions often obscure the range of possible solutions from the designer by only exhibiting a final, "best" solution. This not only deprives the designer of seeing alternatives, but also tends to promote designer fixation on the single, machine-proffered solution (Purcell and Gero 1996). Naturally, optimization methods can only respond to the objective parameters that are coded into the problem, and as a result, noncoded parameters, such as aesthetics, historic context, or meaning, are left out of the optimization process, and ultimately left out of the final design solution. Lost too, in most computer optimization, is the variety of information and creative stimulation that comes from manipulating physical models. Genetic algorithms (GAs) are able to overcome some of the shortcomings that hinder traditional computer-based optimization techniques in application as design tools. They can work to avoid designer fixation by presenting the designer with populations or successive generations of solutions rather than a single end result. By including interactive user selection, GAs can also incorporate noncoded design criteria into the search process.

The idea of generating design forms or shapes using GAs is not new. Many people are familiar with Richard Dawkins's Biomorphs, described in his book *The Blind Watchmaker* (1986) or the graphic art of Karl Sims and William Latham (Kelly 1995). In the area of engineering, form is closely coupled with performance. John Frazer is one who has used GAs in an engineering design sense. In an application dealing with sailing yacht design, the GA optimization considered both objective engineering parameters (stability, center of buoyancy, wetted surface area, prismatic coefficient, blocking), as well as designer criteria such as aesthetic appearance, historic tradition, allusion of form, and so on (Frazer 1996). The Intelligent Genetic Design Tool (IGDT), described in this chapter, develops many of these same ideas.

12.2 ANALYSIS TOOLS VS. DESIGN TOOLS

In recent years a variety of computer-based tools have been developed for the field of architectural engineering and design. Although originally applied to areas of computationally intensive *analysis,* with the ever-increasing size and speed of processors, attempts have been made to develop *design* oriented applications as well. In the field of architecture engineering, the requirements of design tools are distinctly different from those of analysis tools. The analysis process can usually be executed in a regular, predetermined sequence of steps. The sequence may be iterative or have various paths based on the particulars of the problem, but there remains one "correct" solution for one stated problem. For example, for a given structural member, with a given load, a particular analysis will yield one solution for the stresses at some point. The analysis is composed of a sequence of prescribed steps that lead to the solution. Since the analysis is meant to reflect some single existing condition, the solution to the degree that it is correct reflects this same single condition. To the degree that different individuals or different methods show a discrepancy in their solutions, it is assumed that some error is present. In fact, an analysis is usually verified by showing consistency with the solution obtained in another way.

In contrast, the design process is not expected to consistently yield the same solution. Although a designer may follow a sequence of steps, the steps are not self-contained, but influenced by factors outside the process itself (the unique background of the designer, stimuli of the environment, etc.). For a given design problem with a given set of parameters, it is certainly *not* expected that any two designers will come to the same solution. One need only look at the results of any design competition to see the variety of solutions that can be proposed. If a competition for a bridge or building resulted in every entry being identical, the effort would be considered a failure. Design implies creative thinking, and creative thinking does not fit a prescribed set of serial steps. Unlike the analysis of a single given condition, a design solution partially defines the condition and, therefore, seldom results in only one possible solution.

With the advent of parallel processing it was supposed by some that computers would be more capable in the area of design as compared with the earlier serial processing machines. After all, being able to hold two ideas simultaneously is a prerequisite to using metaphors, and metaphors are closely linked to creative thinking (Gordon 1961). But, although parallel processing can shorten computing time, it has not had much impact on enhancing the design capabilities of computer tools. This is because design involves not so much parallel thinking as "lateral thinking." Even work in the area of artificial intelligence has had difficulty in showing capabilities of lateral thinking. For this reason computerized tools have found more success in the area of analysis, which is more easily described as a set procedure.

An IGDT goes beyond a set procedure of analysis to aid the designer in exploring form-finding problems in a creative way. Unlike analysis tools it is not intended to yield one correct solution, but rather to supply the designer with stimulating, plausible directions to consider. An IGDT is intended to be used in the early, form-finding phases of a design problem. As such, it deliberately avoids leading the designer to a single "best" solution, but instead follows the designer's lead in exploring the design space.

12.3 ADVANTAGES OF EVOLUTIONARY SYSTEMS IN DESIGN

In assisting with conceptual design, evolutionary systems, and in particular GAs, contain certain critical characteristics that are lacking in other numerical methods. These are each discussed in the succeeding subsections.

12.3.1 Use of Populations

Fundamental to a GA is the use of populations of solutions rather than a single best solution. This is compatible with the way ideas are often generated and regarded in early design phases. Multiple, simultaneous ideas are necessary for the dynamic movement of thought associated with creative design. Ideas play against each other, leading to further new ideas. As was stated earlier, creative design is typically not a single-path, linear process, but a more complex, multipath exploration of many possible ideas.

12.3.2 Recombination and Mutation

A GA uses recombination and mutation to generate new solutions to a problem. This is very similar to design tools developed by William Gordon (1961) that attempt to combine diverse aspects of different solutions to achieve a new perspective. In a GA, groups of possible solutions are recombined in a like manner. One of Gordon's techniques is to fragment and recombine words and phrases. In synectics one of the "operational mechanisms" suggested to promote creative thinking is to play with words and phrases and their meanings as a way of making the familiar strange. Similarly, Albert Einstein comments that "combinatory play seems to be the essential feature in productive thought" (Reisner 1931). This "play" is similar to the mechanisms of random mutation and recombination used in genetic techniques.

12.3.3 Wide Search of Design Space

Through the genetic operators of mutation and random recombination, GAs are very effective at exploring the complete design space of a problem. Complex problems may contain many regions of near-optimal solutions. Rather than focusing on just one such region, GAs will attempt to explore a range of such regions simultaneously. In reference to creative design, George Prince (1970) stresses the importance of free speculation as a means of enhancing creative thought:

> We believe that as the expert accumulates the specific knowledge that makes him so valuable he also incorporates the accepted certainties that are not really certain. This explains why, historically, so many innovative breakthroughs have come from outsiders rather than from those who are thought to be most expert in the particular field.

Using mechanisms of random mutation, GAs are able to retain a degree of speculation. In this way they can help even Prince's "expert" to consider new possibilities in the design space.

12.3.4 No Knowledge of the Objective Function

The fact that the GA itself operates independently of the objective function allows it to be controlled, without programming, by the designer. The GA is steered by means of a fitness function, but the source of that fitness is not necessarily defined by coded objectives alone. The GA can be guided by any means of evaluation, either coded, noncoded, or a combination of both. This is very important when considering some hard-to-code, qualitative or subjective design criteria such as aesthetics. Engineering design combines subjective and objective design criteria. The IGDT works on both levels at once, using coded objective criteria in searching for good solutions, and noncoded subjective criteria supplied by the designer's interactive selections.

12.3.5 Imitation of Human Design Process

A GA learns in a way that is analogous to human learning modes. It seeks a solution by considering many options. Like the near-random flow of ideas that a designer will sift through at the start of each project in search of usable concepts, a GA processes thousands of solutions, comparing, crossing, recombining, altering, sorting, keeping the best, and always scanning the design space for better ideas. The process is not always direct, but it is always goal oriented. It is also thinkable that the criteria may evolve as the project matures. As the orienting fitness function changes, the GA automatically adapts and searches in the shifted direction.

12.3.6 Can Learn from Designer

Like the adaptation of biological systems to their environment, the GA can learn from the fitness information it is fed regardless of the source. Thus, the GA can be guided with criteria that even the designer might find difficult to express. In so far as the designer is consistent in ranking the fitness of solutions, the GA will learn from the designer. By considering the solutions proposed by the GA, the designer too may learn and form a new viewpoint of proposed design solutions. The GA is also able to adapt to changes in the designer's understanding without direct reprogramming from the designer.

12.4 CHARACTERISTICS OF AN IGDT

An Intelligent Genetic Design Tool (IGDT) is a tool that is able to dynamically adapt to evolving design criteria, through interaction with the designer, to aid in the exploration of a range of good solutions. It is intended to assist the designer in the early, conceptual design phases.

12.4.1 Definition of the IGDT Concept

The concept of an IGDT is significantly different from traditional analysis and design programs in three ways:

- It does not rely solely on preprogrammed, quantifiable objectives.
- It provides an intelligent interface, responsive to the user's ideas.
- It aids and stimulates the creative exploitation of the design space.

Although an IGDT uses optimization, it is distinguished from traditional optimization methods in its ability to adapt to nonprogrammed fitness criteria learned from user interaction. An IGDT is able to learn by employing genetic operators of recombination, mutation, and selection to a population of solutions that evolve based on the selective pressure that can be supplied through human interaction. Because the objectives of an IGDT are not explicitly preprogrammed, an IGDT can be implemented in earlier design phases than is possible with more common analysis tools, without the danger of causing design fixation or prematurely restricting the design process. Many initial design considerations that are qualitative parameters, such as aesthetics or complexity of form, have traditionally been defined through examples. An IGDT, which communicates with the user through examples, provides a more natural and effective tool for design than programs that require predefined, quantifiable objectives. Also, because an IGDT learns from interaction with the user, it is more easily approached by non-computer-oriented users. The concept of an IGDT is applicable to many fields involving design.

Architectural design problems require solutions that consider a wider spectrum of parameters than is ordinarily covered by base functionality and cost. To consider qualitative parameters like aesthetics or complexity of form, a design tool must remain flexible and have the ability to adapt to criteria learned from the user. Although in recent years several "computer-aided design tools" have been developed, these tools find application primarily in the later design phases. By offering single "optimized" solutions to the initial parameters used in early conceptual design phases, this type of tool can actually hinder the designer's creative exploration of the design space. The temptation to the designer is to accept the direction offered by the optimization analysis without sufficient exploration of alternatives. Simply having a complete solution presented may lead to design fixation where the presence of one idea tends to block other ideas from being considered.

"Not preprogrammed" means that the objective function or fitness criteria are determined or altered as the tool is being used by the designer. In traditional optimization programs the fitness or objective function is determined in advance, and the solution converges toward a solution that optimizes this preprogrammed criteria. In an IGDT the final design criteria are supplied by the user while the program is being run in the form of selection or ranking of individual proposed solutions. These fitness criteria, in the form of ranking by the designer, are not preprogrammed. For example the designer may have several qualitative criteria based on concepts of aesthetics, practicality of construction, limits of time, space, skill, and so on. In a real application the designer may have many such criteria that overlap or even conflict. The designer may not even be able to completely verbalize such criteria in words let alone computer code. Nevertheless, in so far as the designer is consistent with the selection or ranking of individual solutions based on this personal criteria, the IGDT will function as a useful tool.

"Intelligent" means that an IGDT both learns from the designer and anticipates the designer's direction in the exploration of design spaces. This allows an IGDT to make intelligent, rather than simply random, proposals back to the designer. The criteria used, particularly in early design phases, are dynamic. An IGDT allows for the adaptation of the design criteria as well as the design results. An IGDT is able to adapt by following implicit criteria learned from the selections made by the designer. It adapts dynamically to the designer's direction, even as that direction may be evolving over the course of a session. This ability to adapt to changing criteria within a session is absolutely necessary for a design tool. It is one of the characteristics that distinguishes design from analysis. By providing the designer with a variety of different solutions, the IGDT stimulates the designer's creativity. On the other hand, an analysis tool can only answer questions put to it by the designer. Analysis tools imply a single "correct" or best solution. They are convergent and thereby restrictive to creativity. Since the guiding criteria of an IGDT are learned through interaction with the designer, the IGDT can adapt to these evolving criteria without forcing the designer in a predetermined direction.

"Exploitative" means that an IGDT not only searches a design space for optimal solutions, but it can actually alter or explore the solutions that might be found in new design spaces. By allowing the design criteria to be altered, an IGDT can be used to search variations of the problem criteria (alternate design spaces) for solutions that better satisfy the goals. Traditional methods with predefined objective functions have as a goal the discovery of the single "optimal" solution. An IGDT seeks *populations* of good solutions with a significant degree of difference. The concept of an IGDT is not only to search for the best solutions defined in one design space, but to explore different possible design spaces in a way that is helpful and stimulating to the designer. It offers both solutions that tend in the direction of the designer's criteria, as well as solutions that explore new directions defined by alterations to the criteria. In this way the IGDT is intelligent in that it is able to offer possibly good solutions in a direction that may not have been previously foreseen. The designer interacting with the IGDT may recognize some such new directions as having potential, and through selection of these solutions allow the IGDT to explore the new design space further. In this way a dialogue exists between the IGDT and the designer in which both share in the exploration of solutions.

12.4.2 Relation of IGDT to Design Process

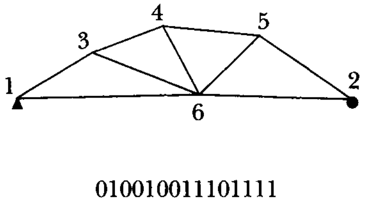
The conceptual design process is iterative and usually contains a certain amount of wandering. The path toward the final solution is not always direct nor constantly progressive. Tools that are highly directive, which seek a single best solution, can actually hinder design by narrowing the scope of consideration too early in the process, before sufficient possibilities have been explored. People who teach design are very familiar with the tendency students will have to latch onto a solution to the extent of refusing to justly consider other possibilities (Koberg and Bagnall 1976). In the field of psychology this is called "fixation" or "being fixated on a solution." A tool that offers a single "best" solution runs the danger of causing fixation and thus setting up a mental block that actually hinders the designer from considering other solutions. For this reason, tools that may be excellent analysis aids, may be very poor design aids. An IGDT is intended to be compatible with the design process. It allows a certain amount of wandering in exploring the different design spaces. It acts as a tool in the hand of the designer. It follows the designer's direction and responds to the designer's evolving criteria. It is also an intelligent tool in that having learned the designer's personal criteria, it is able to propose other different solutions that meet those criteria. The designer enters into a dialog with the tool. The IGDT makes reasonable proposals, and the designer gives comment to the proposals in the form of selection, alteration, or further proposals. The IGDT learns from the interaction and makes further new proposals. In addition the IGDT searches for other reasonable solutions and includes them in the next round of proposals. This pattern is similar to the way two designers interact in exploring options for a design. There is a certain amount of trading of ideas, as well as a certain amount of individual suggestion. The suggestions from the one may trigger new ideas and suggestions from the other. Well-known techniques, such as speech manipulation or Gordon's synectics (Gordon 1961), are attempts to generate new and creative solutions to a problem. An IGDT fulfills this same role as an idea generator. The genetic operators of recombination and mutation help to explore design spaces by generating new solutions that respond to the considerations of the designer. Because an IGDT always works with a population of solutions, there is less danger that a premature answer will be accepted. An IGDT tends to be expansive rather than constrictive in the design process.

GAs are used as the basis for an IGDT because they are conceptually very close in operation to the way many designers work. GAs generate populations of individuals in the way that designers create a pool of ideas to draw from. GAs operate on this population by recombining parts of different individuals or altering existing individuals though mutation (Holland 1975). Designers, too, combine good aspects of various ideas, and alter old ideas to fit new situations. GAs search a wide design space for various good solutions before narrowing the scope to a single best solution. Designers behave similarly by considering several options. Also, because a GA need not have programmed knowledge of the solution objective, the fitness of the individual solutions can be determined by the designer using qualitative criteria. By not requiring the design criteria to be numerically expressed or directly programmed, an IGDT remains flexible and responsive to changes dictated by the designer.

For design applications involving form, the ability to view and manipulate proposed solutions is essential to the decision-making process. In this chapter an IGDT is demonstrated using a trussed structural system as an example. The concept presented, however, is not limited to trusses, or even structural systems. The basic concept of an IGDT can be adapted to a wide range of form-determining design problems.

12.5 MECHANICS OF AN IGDT

The IGDT functions in two different layers. In the outer layer, the IGDT interacts directly with the user. It proffers solutions to the designer for critique and



\	1	2	3	4	5	6
1	\	0	1	0	0	1
2		\	0	0	1	1
3			\	1	0	1
4				\	1	1
5					\	1
6						\

12.1 FIGURE An example truss with numbered nodes, incidence matrix, and defining binary string.

ranking. The designer's ranking provides the fitness value for the individual solutions.

A second, inner layer operates within the outer layer. The inner layer searches the design space for good solutions to feed to the outer layer. While the outer layer searches for acceptable topologies with the designer, the inner layer finds optimal geometries for each topology. In this way the proffered solutions are reasonably good solutions for the selected topologies.

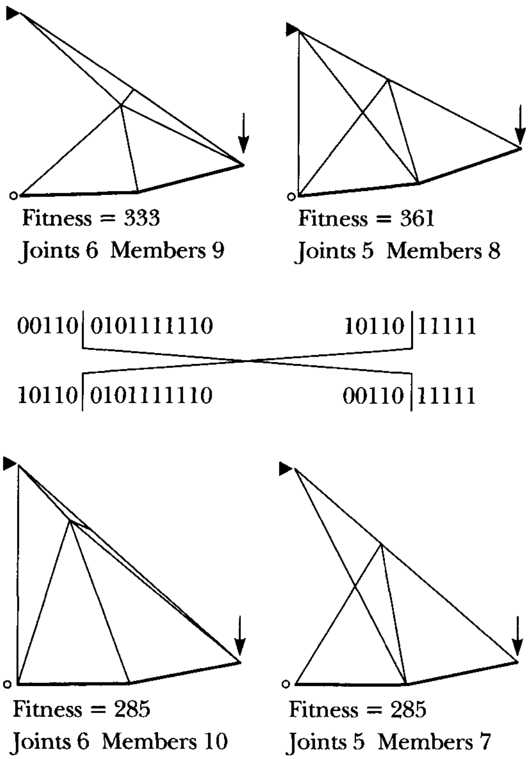
The outer layer uses the structure's incidence matrix as a chromosome. The incidence matrix describes the connectivity of the members and joints by recording the start and end joint number of each member. As shown in Figure 12.1, the incidence matrix can easily be converted to a binary string. The member numbering normally contained in the matrix can be defined to follow a simple ordinal sequence.

In this form the matrix lends itself well for use as a chromosome in a conventional binary string GA, that is, one patterned after Holland and Goldberg (Holland 1975; Goldberg 1989). With the intent of providing less disruption and of preserving longer schemata, single-point crossover is used. At this point no testing has been performed to bear this out. Figure 12.2 shows an example of the topology breeding procedure.

Not every topology produced randomly or by crossover represents a stable, usable truss. For example, joints may be left with no connecting members, or members may be left isolated with no connection to the rest of the structure. These cases are simply regarded as stillbirths, and the process that produced them is repeated until a stable solution is found.

The geometry GA works with a given topology to find good geometry solutions. The geometry is defined by the Cartesian coordinates (real numbers) of each joint in the structure.

The programming mechanics used for the inner layer are based on the CHC method developed by Eshelman and Schaffer at Philips Laboratories (Eshelman

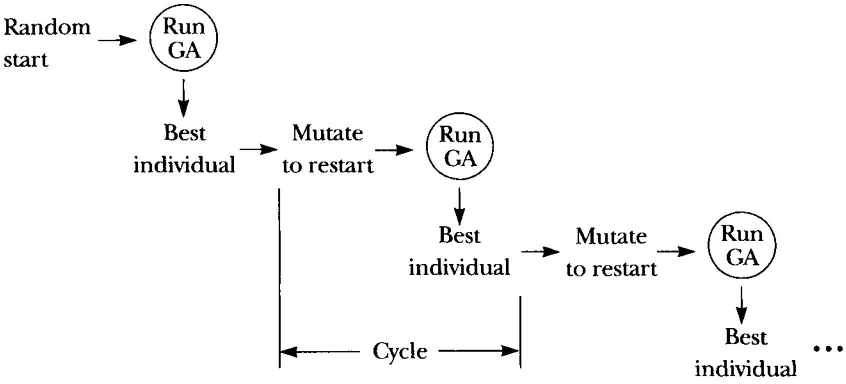

Fitness = 333
Joints 6 Members 9
Fitness = 361
Joints 5 Members 8
00110 0101111110
10110 11111
10110 0101111110
00110 11111
Fitness = 285
Joints 6 Members 10
Fitness = 285
Joints 5 Members 7

12.2 FIGURE

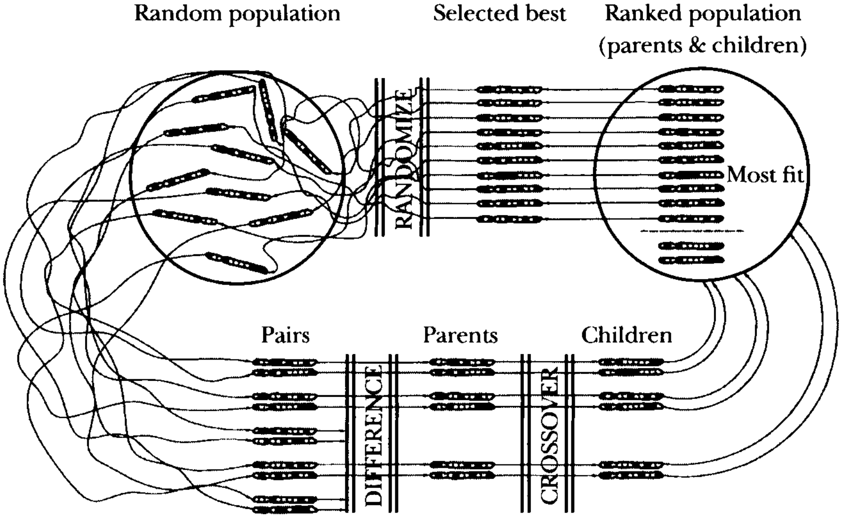
The breeding of two different topologies using the incidence matrices.

1991). The CHC varies from the more traditional GA methods in that it uses a strongly elitist, steady-state selection strategy and segregates the operations of mutation and crossover. CHC has the advantage of being able to work with relatively small populations of 50 individuals per generation and still succeed over a wide range of problems. The breeding algorithm used is similar to that employed in a ($\mu + \lambda$)-evolution strategy (Bäck, Hoffmeister, and Schwefel 1992). CHC is also unique in that it segregates mutation and crossover by running in cycles using crossover only and then restarting a cycle by mutating the best individual to fill the restart population. Figure 12.3 shows the overall CHC cycling, and Figure 12.4 shows the GA portion of one cycle. Figure 12.5 shows six generations from a sample run.

Because the joint coordinates are real numbers with a meaningful spatial location, it was decided to incorporate these qualities into the breeding mechanism. During breeding, the joint coordinates of a child are selected from points


Random start
Run GA
Best individual
Mutate to restart
Run GA
Best individual
Mutate to restart
Run GA
Best individual
...
Cycle

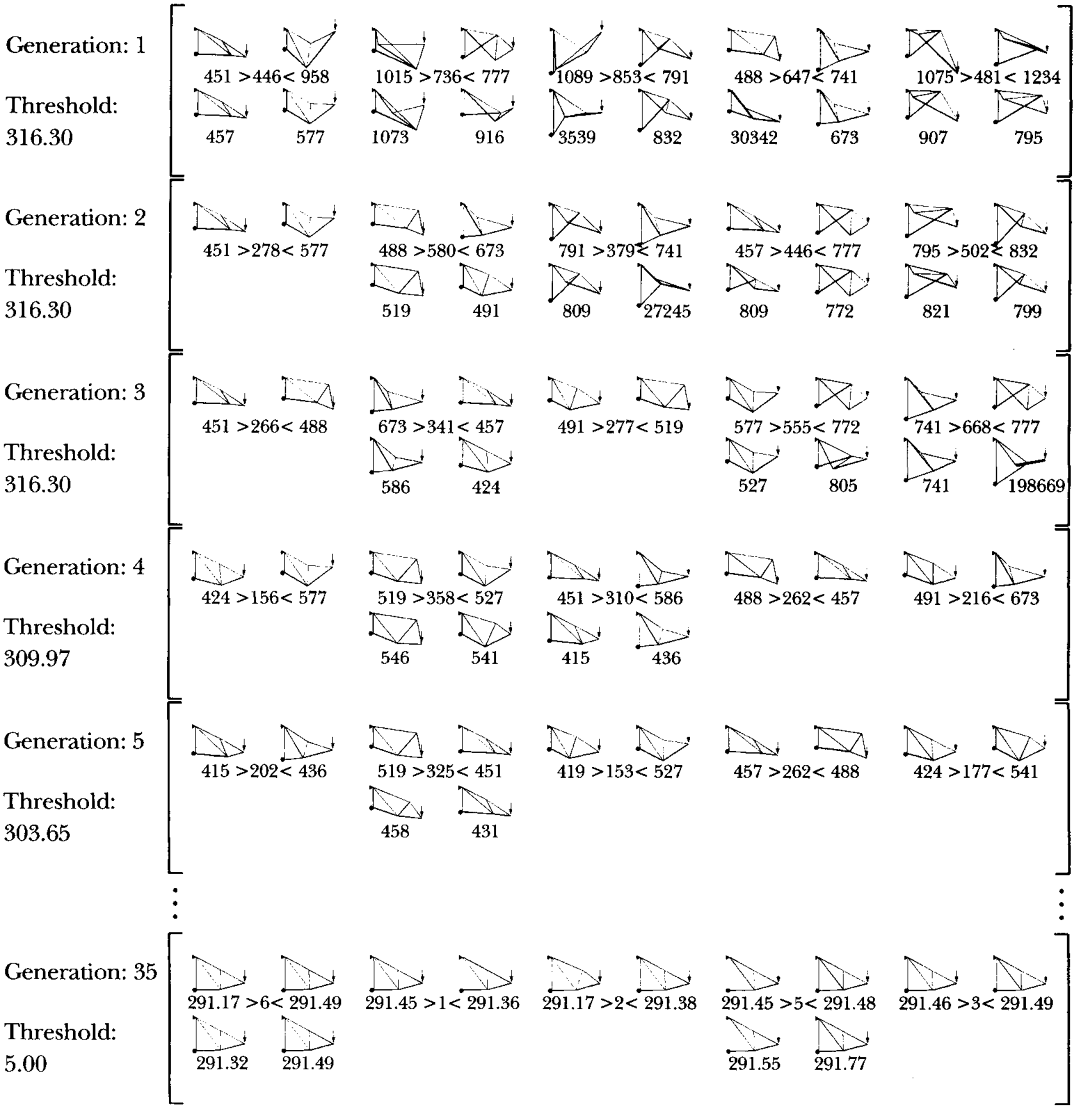
12.3 FIGURE Overall plan of the CHC GA.


Random population
Selected best
Ranked population (parents & children)
RANDOMIZE
Most fit
Pairs
Parents
Children
DIFFERENCE
CROSSOVER

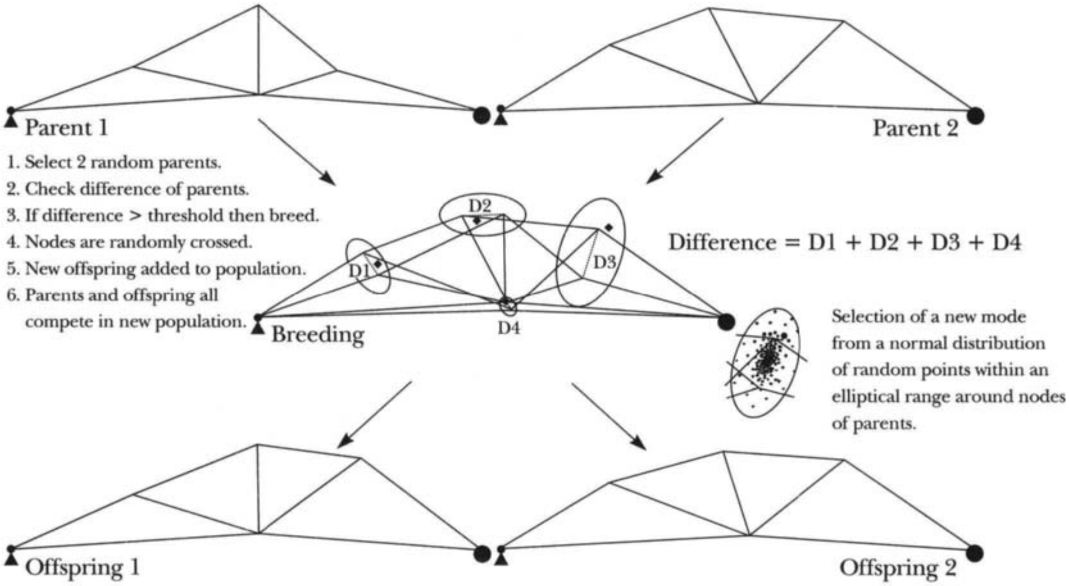
12.4 FIGURE The "GA" portion of the CHC.

in a natural distribution about the parent points. Figure 12.6 shows an example of the breeding process for two trusses.

Although other optimization methods could be substituted in place of the CHC method just described, the method chosen gave good results and is easily adapted to a wide range of problems and objectives.


Generation: 1
Threshold:
316.30
451 >446< 958
1015 >736< 777
1089 >853< 791
488 >647< 741
1075 >481< 1234
457
577
1073
916
3539
832
30342
673
907
795
Generation: 2
Threshold:
316.30
451 >278< 577
488 >580< 673
791 >379< 741
457 >446< 777
795 >502< 832
519
491
809
27245
809
772
821
799
Generation: 3
Threshold:
316.30
451 >266< 488
673 >341< 457
491 >277< 519
577 >555< 772
741 >668< 777
586
424
527
805
741
198669
Generation: 4
Threshold:
309.97
424 >156< 577
519 >358< 527
451 >310< 586
488 >262< 457
491 >216< 673
546
541
415
436
Generation: 5
Threshold:
303.65
415 >202< 436
519 >325< 451
419 >153< 527
457 >262< 488
424 >177< 541
458
431
Generation: 35
Threshold:
5.00
291.17 >6< 291.49
291.45 >1< 291.36
291.17 >2< 291.38
291.45 >5< 291.48
291.46 >3< 291.49
291.32
291.49
291.55
291.77

12.5 FIGURE A sample of six generations from a CHC GA.


Parent 1
Parent 2
1. Select 2 random parents.
2. Check difference of parents.
3. If difference > threshold then breed.
4. Nodes are randomly crossed.
5. New offspring added to population.
6. Parents and offspring all compete in new population.
D1
D2
D3
D4
Breeding
Difference = D1 + D2 + D3 + D4
Selection of a new mode from a normal distribution of random points within an elliptical range around nodes of parents.
Offspring 1
Offspring 2

12.6 FIGURE The breeding of two parents and the calculation of a difference threshold.

12.6 IGDT OPERATION

The operation of the IGDT is iterative and can continue as long as the designer finds it productive. Like any dialogue, it will reach a point of stability where so little new ground is being covered that continuing is not effective. Because the IGDT interacts intelligently with the designer, repeated sessions at later times can offer different results since the designer's own understanding of the criteria will evolve over time. Therefore, more complex problems may benefit from multiple sessions spaced a few days apart. Within one session the activity can be outlined as described in the succeeding five subsections.

12.6.1 Problem Definition

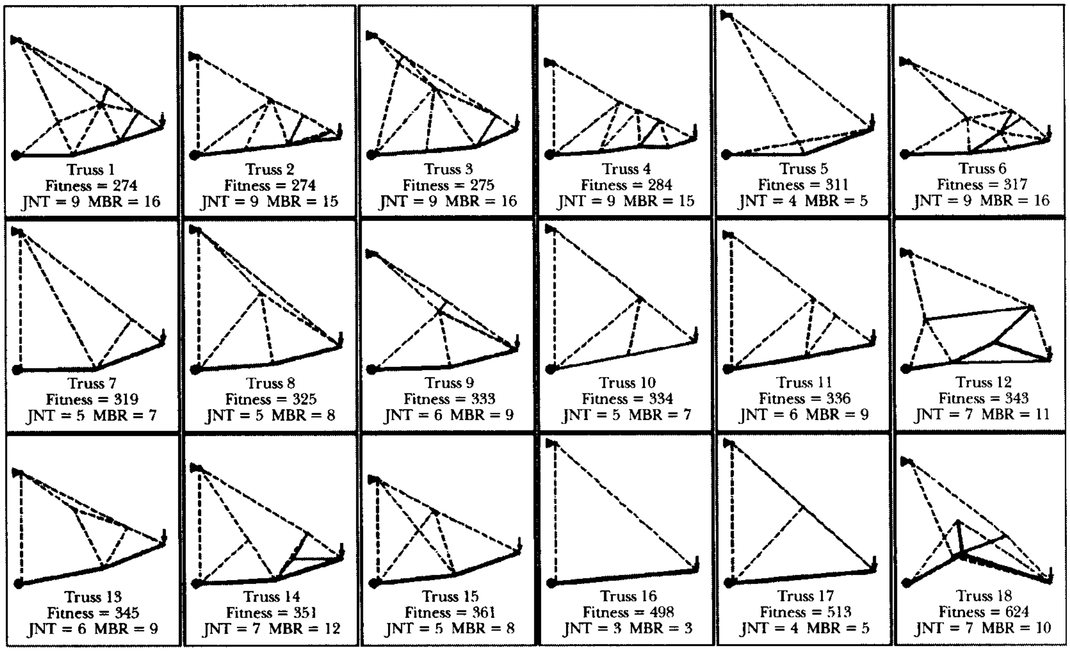
The user initially sets constant design parameters that are stored in a data file read by the IGDT at the start of a session. Constant criteria include material constants and properties; topology constants such as symmetry; geometry constants such as support positions or required load points; required load cases and load combinations; analysis method; and output specifications. For the cantilever truss example the following problem definition was used:

- *System:* Plane truss
- *Material:* Steel (ASTM A-36)
- *Sections:* Pipe (based on Schedule 40 with continuous size range)
- *Supports:* One with fixed x- and y-coordinates (upper support) and one with fixed x-coordinate only (the lower support)
- *Loads:* 1-point load of 10,000 lbs at 30 feet from the supports
- *Analysis:* AISC ASD steel code
- *Fitness:* Least weight

The program is modularly designed to allow for the easy incorporation of a variety of optional subroutines. The optional subroutines are also chosen at the start of the trial. In the current truss design program, optional subroutines include material (steel or wood), analysis methods (wood or steel codes, Euler buckling, or simple axial stress), loading types, and combinations (point loads or self-weight). Any joint coordinates can be set to appear positionally "frozen" to the GA. They can be given a prespecified, set position in the x or y direction or both directions. This is particularly useful in defining supports or specifically located point loads. Fitness in this example was limited for simplicity to weight, but it is possible to include other criteria as well, such as total number of joints, complexity of joints (number of members meeting at a joint), oversize members, waste from standard lengths, and so on.

12.6.2 Initial IGDT Generation

Figure 12.7 shows the first generation of solutions from the IGDT. These represent either different topologies and/or significantly different geometries. The dashed members indicate tension, and continuous members indicate compression. In this example a population of 18 was used. This is seen as an overseeable number for the designer. Each solution is shown in a box, and they are all sorted from top to bottom by fitness. The solutions are numbered consecutively from the first generation onward and can be retrieved from storage at any time. The proposals reflect the inner layer optimization based on the fitness criteria set by the designer, but are intended to display a variety of different good solutions rather than just the best solutions.



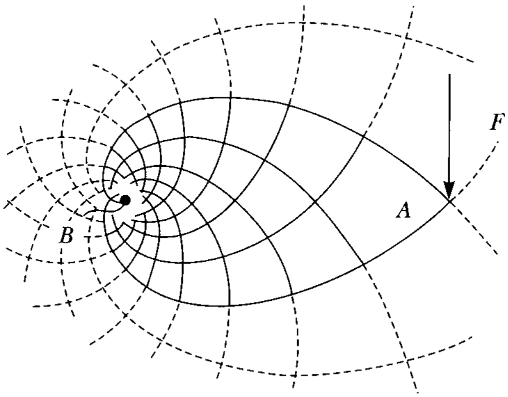
Truss 1 Fitness = 274 JNT = 9 MBR = 16
Truss 2 Fitness = 274 JNT = 9 MBR = 15
Truss 3 Fitness = 275 JNT = 9 MBR = 16
Truss 4 Fitness = 284 JNT = 9 MBR = 15
Truss 5 Fitness = 311 JNT = 4 MBR = 5
Truss 6 Fitness = 317 JNT = 9 MBR = 16
Truss 7 Fitness = 319 JNT = 5 MBR = 7
Truss 8 Fitness = 325 JNT = 5 MBR = 8
Truss 9 Fitness = 333 JNT = 6 MBR = 9
Truss 10 Fitness = 334 JNT = 5 MBR = 7
Truss 11 Fitness = 336 JNT = 6 MBR = 9
Truss 12 Fitness = 343 JNT = 7 MBR = 11
Truss 13 Fitness = 345 JNT = 6 MBR = 9
Truss 14 Fitness = 351 JNT = 7 MBR = 12
Truss 15 Fitness = 361 JNT = 5 MBR = 8
Truss 16 Fitness = 498 JNT = 3 MBR = 3
Truss 17 Fitness = 513 JNT = 4 MBR = 5
Truss 18 Fitness = 624 JNT = 7 MBR = 10

12.7 FIGURE An initial IGDT generation with proposed solutions arranged by the specified fitness of least weight.

12.6.3 Initial Generation with Designer Selection/Interaction

Presented with a palette of proposals from the IGDT, the designer begins the iterative dialogue that allows the IGDT to learn the preferences of the individual designer on the specific project being investigated. The designer proceeds by selecting and marking best and worst proposals.

Figure 12.8 shows a Mitchell diagram for an "optimum" solution of a cantilever. Mitchell's support conditions are somewhat different, and he did not consider stability (no buckling). Nonetheless, Truss 1 is very similar in form to this idealized solution. But, for the sake of the example, it will be supposed that the designer sees visual and practical problems with this form. From the front view (looking from the loaded point, back toward the supports) the bottom chords of the structure are exposed, which might cause weathering problems. Further it will be supposed that the designer desires the visual effect of a clean line in elevation view at the point load with preferably an upwardly curving underside. The



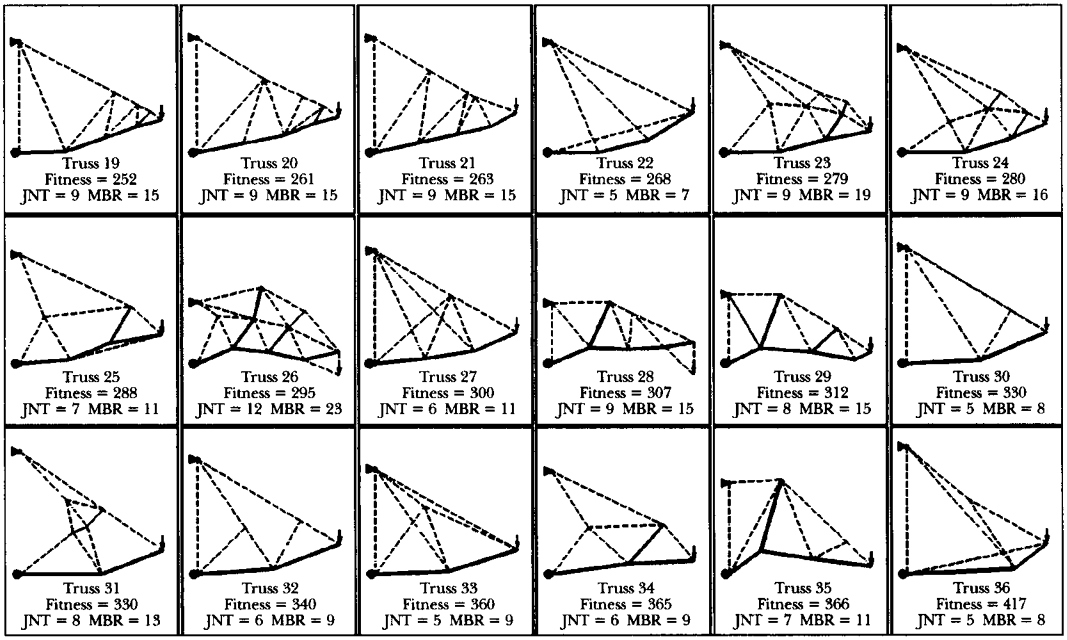
12.8 FIGURE Mitchell's diagram for a cantilever case (Mitchell 1904).

designer indicates these preferences by selecting Trusses 2, 4, 12, and 18. The designer considers solutions 10, 11, 16, and 17 too simple visually and marks them for deletion. The effect of this selection is to remove the undesirable trusses form the breeding pool, and to increase the likelihood of the selected trusses to breed more often.

12.6.4 Second-Generation IGDT Response

Using the designer's input, the IGDT searches for a new set of proposals. First, new topologies are found using binary GA operations of mutation and recombination performed on the incidence chromosome shown in Figures 12.1 and 12.2. The children topologies are then established with good geometries using the CHC GA.

Again the proposals represent either different topologies and/or significantly different geometries and are sorted and numbered by fitness. Figure 12.9 shows the IGDT's response to the designer. Unlike the initial proposals of the IGDT, all subsequent generations of proposals are derived from, or influenced by, the interaction with the designer. Some of the same topologies show up, but as noted above, parents are not included in subsequent generations, unless deliberately retrieved by the designer and reintroduced to the breeding population. Upon examining Figure 12.9, one sees that the form tends naturally toward the form of the Mitchell diagram, but several geometries are now proposed with concave bottom chords.


Truss 19
Fitness = 252
JNT = 9 MBR = 15
Truss 20
Fitness = 261
JNT = 9 MBR = 15
Truss 21
Fitness = 263
JNT = 9 MBR = 15
Truss 22
Fitness = 268
JNT = 5 MBR = 7
Truss 23
Fitness = 279
JNT = 9 MBR = 19
Truss 24
Fitness = 280
JNT = 9 MBR = 16
Truss 25
Fitness = 288
JNT = 7 MBR = 11
Truss 26
Fitness = 295
JNT = 12 MBR = 23
Truss 27
Fitness = 300
JNT = 6 MBR = 11
Truss 28
Fitness = 307
JNT = 9 MBR = 15
Truss 29
Fitness = 312
JNT = 8 MBR = 15
Truss 30
Fitness = 330
JNT = 5 MBR = 8
Truss 31
Fitness = 330
JNT = 8 MBR = 13
Truss 32
Fitness = 340
JNT = 6 MBR = 9
Truss 33
Fitness = 360
JNT = 5 MBR = 9
Truss 34
Fitness = 365
JNT = 6 MBR = 9
Truss 35
Fitness = 366
JNT = 7 MBR = 11
Truss 36
Fitness = 417
JNT = 5 MBR = 8

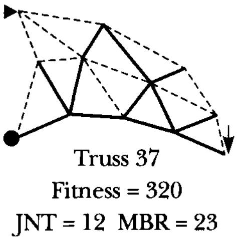
12.9 FIGURE The second generation produced by the IGDT.

12.6.5 Second-Generation Designer Interaction

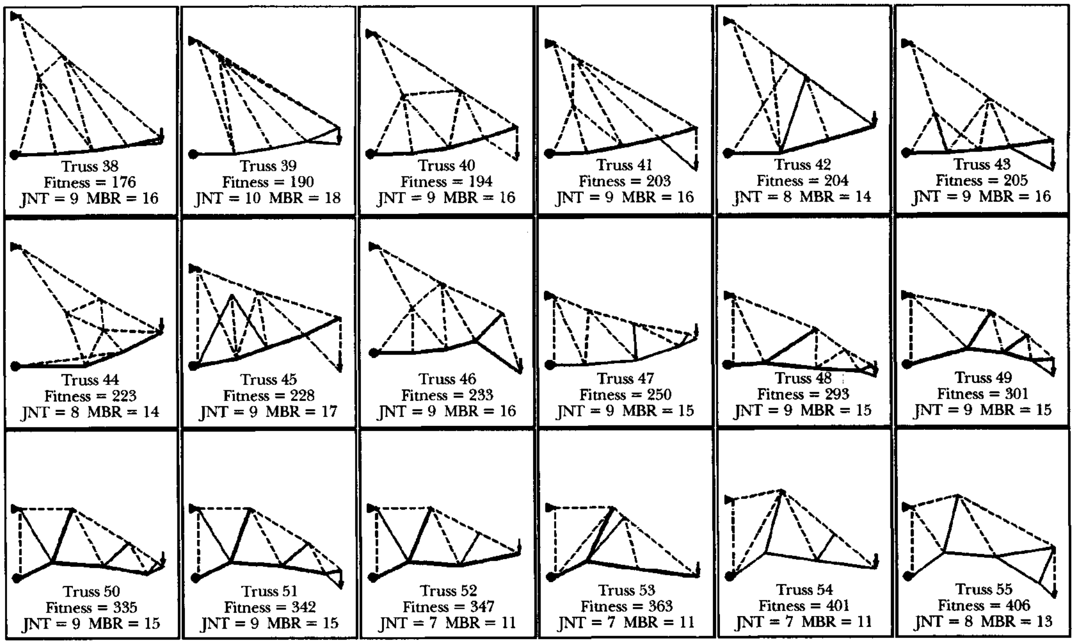
Again the designer makes selections of desired results. Each occurrence of a geometry with concave bottom chords is again consequently selected. Trusses that are not selected and not deleted remain in the breeding population but with no enhanced probability of breeding. Also, the designer is free to add self-made (mutated) topologies or geometries to the breeding pool, or recall individuals from previous generations. In this way the designer can share in directing the outcome. In this example the designer added a variant of Truss 26 to the population. The added truss is shown in Figure 12.10.

12.6.6 Third Generation

Figure 12.11 shows the IGDT's response to the selections made in the previous generation. There are now several new topologies that exhibit geometries with


Truss 37
Fitness = 320
JNT = 12 MBR = 23

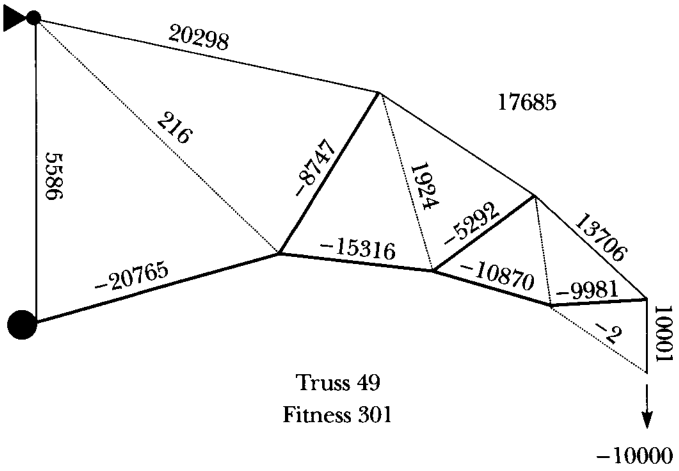
12.10 FIGURE Designer-modified Truss 26, introduced as Truss 37.


Truss 38
Fitness = 176
JNT = 9 MBR = 16
Truss 39
Fitness = 190
JNT = 10 MBR = 18
Truss 40
Fitness = 194
JNT = 9 MBR = 16
Truss 41
Fitness = 203
JNT = 9 MBR = 16
Truss 42
Fitness = 204
JNT = 8 MBR = 14
Truss 43
Fitness = 205
JNT = 9 MBR = 16
Truss 44
Fitness = 223
JNT = 8 MBR = 14
Truss 45
Fitness = 228
JNT = 9 MBR = 17
Truss 46
Fitness = 233
JNT = 9 MBR = 16
Truss 47
Fitness = 250
JNT = 9 MBR = 15
Truss 48
Fitness = 293
JNT = 9 MBR = 15
Truss 49
Fitness = 301
JNT = 9 MBR = 15
Truss 50
Fitness = 335
JNT = 9 MBR = 15
Truss 51
Fitness = 342
JNT = 9 MBR = 15
Truss 52
Fitness = 347
JNT = 7 MBR = 11
Truss 53
Fitness = 363
JNT = 7 MBR = 11
Truss 54
Fitness = 401
JNT = 7 MBR = 11
Truss 55
Fitness = 406
JNT = 8 MBR = 13

12.11 FIGURE The third generation produced by the IGDT.

concave bottom chords. The process continues as with the previous generation. The designer indicates positive and negative preferences, and the IGDT breeds the surviving solutions to find new possible solutions that lie in the direction of the designer's preferences.

Even at this early stage some of the IGDT proposals can be helpful to the designer. For example, Truss 49 from the third generation (enlarged in Figure



12.12 FIGURE The final solution selected by the designer.

12.12) is in the direction being pursued by the designer. It is similar to the designer's own proposal (Truss 37 in Figure 12.10) but offers some advantages of simplicity and economy of material.

The most weight-efficient solution found by the IGDT (Truss 38 in Figure 12.11) resembles a ridged bottom plate hung in place with tension members. This may indicate a potentially new concept for the designer to consider as opposed to a completely ridged truss. Thus, the designer can continue to explore possible design concepts with the IGDT. Like any design process, it ends when the designer is satisfied with the solution (or constraints such as time force a decision). In the course of the exchange, the IGDT learns the designer's preferences, or whatever those preferences may be based on. The designer also learns as the range of possible solutions is explored. As new sets of proposals are presented, the images will suggest new considerations and concepts to the designer. In the course of a session, it is expected that the thinking and considerations made by the designer in responding to the IGDT will undergo a development of their own. This is precisely why genetic algorithms work so well in the IGDT. Since GAs do not require any knowledge as to how the selection decisions are made, they are free to follow whatever direction is indicated by the designer. As a result, the path toward the solution will not be direct. Old solutions may from time to time resurface under changing criteria. This is not important. What is important is that the design space be thoroughly explored and made apparent to the designer. The success of the IGDT is based more on a thorough search of possibilities than on finding precisely the single, ultimate best solution.

12.7 CONCLUSIONS

Although it would be easy to suppose that an IGDT helps the designer to find a solution that meets the aesthetic and performance criteria while still providing good economy with low weight, this chapter does not really try to prove the point. The tool is intended to help the designer in the exploration of structures that are otherwise difficult to assess. A simple, flat cantilever truss is not the best example of this, but is an example that lies within the current limits of the program's development. In addition, the "designer" in this case was the same as the program's author, and the author of this chapter—not altogether an impartial test. The problem, too, was chosen as one that worked out to some degree. In most cases there is much more wandering in the procedure. But the wandering need not always be regarded as negative, provided that it does not simply circle the same position.

The form chosen by the designer differed significantly from ideal Mitchell diagram form, which direct optimization programs would suggest. In the process of developing the final solution, many possible directions were explored, and the designer received a good overview of potential solutions before coming to a conclusion. In this way the IGDT acted as a true aid to the designer in finding the most satisfying solution.

ACKNOWLEDGMENTS

I wish to thank Dr. Werner Sobek, director of the Institute for Lightweight Structures at the University of Stuttgart, for his patient support, as well as the University of Tennessee for a grant that aided in this work.

REFERENCES

Bäck, T., R. Hoffmeister, and H.-P. Schwefel (1992). A Survey of Evolution Strategies. In *Proceedings of the Fourth International Conference on Genetic Algorithms,* Morgan Kaufmann.

Dawkins, R. (1986). *The Blind Watchmaker.* Harlow Logman.

Eshelman, L. (1991). The CHC Adaptive Search Algorithm: How to Have Safe Search When Engaging in Nontraditional Genetic Recombination. In *Foundations of Genetic Algorithms,* Morgan Kaufmann.

Frazer, J. (1996). The Dynamic Evolution of Designs. In *4D Dynamics Conference on Design and Research Methodologies for Dynamic Form,* De Montfort University, Leicester, UK.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Gordon, W. J. J. (1961). *Synectics: The Development of Creative Capacity.* Harper & Row.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Kelly, K. (1995). *Out of Control: The New Biology of Machines, Social Systems and the Economic World.* Addison-Wesley.

Koberg, D., and J. Bagnall. (1976). *The Universal Traveler—A Soft-Systems Guide to Creativity, Problem-Solving, and the Process of Reaching Goals.* Morgan Kaufmann.

Mitchell, A. G. M. (1904). The Limits of Economy of Material in Frame-structures. *Philosophical Magazine and Journal of Science* 8(6).

Prince, G. M. (1970). *The Practice of Creativity: A Manual for Dynamic Group Problem Solving.* Harper & Row.

Purcell, T. A., and J. S. Gero. (1996). Design and Other Types of Fixation. *Design Studies* 17(4):363–383.

Reisner, A. (1931). *Albert Einstein.* Thornton Butterworth.

IV PART EVOLUTIONARY ART

13 Eons of Genetically Evolved Algorithmic Images
Steven Rooke

14 Art, Robots, and Evolution As a Tool for Creativity
Luigi Pagliarini, Henrik Hautop Lund

15 Stepping Stones in the Mist
Paul Brown

16 Evolutionary Generation of Faces
Peter J. B. Hancock, Charlie D. Frowd

17 The Escher Evolver: Evolution to the People
A. E. Eiben, R. Nabuurs, I. Booij

We've seen two prime examples of creativity and evolution so far in the previous two parts: music and design. The third is just as crucial to this book: *art.* There can be nothing as unconstrained and ill-defined as art. Nothing so subjective, fuzzy, and difficult to analyze. And yet we can evolve it. This part provides the second major link to the sister book *Evolutionary Design by Computers.* Again we have a part on evolutionary art filled with some of the best practitioners in the field.

We begin with Chapter 13, "Eons of Genetically Evolved Algorithmic Images," by Steven Rooke—one of the best known of the current "evolutionary artists." Steven describes his excitement and illustrates how he has used GP to evolve some astonishing pieces of art.

Louise Sizer-James

Chapter 14, "Art, Robots, and Evolution As a Tool for Creativity," by Luigi Pagliarini and Henrik Hautop Lund of the LEGO Lab, University of Aarhus, describes how technology and art have come together, and what that means for art and artist. They provide examples of systems, including their well-known Artificial Painter and also the use of robots to generate art.

Chapter 15, "Stepping Stones in the Mist," is the third chapter in this part. Paul Brown, an artist, writer, and executive editor of *fineart forum,* describes his own motivation and inspiration as he uses methods such as cellular automata to create his work. Some stunning pieces are included in his chapter.

We then examine a very practical application for the methods created by evolutionary artists. In Chapter 16, "Evolutionary Generation of Faces," Peter Hancock and Charlie Frowd, of Stirling University, show how aesthetic selection by a user can enable the evolution of photorealistic faces.

Finally, we look at how evolution can produce images in the style of Escher, in Chapter 17, "The Escher Evolver: Evolution to the People." Gusz Eiben and his colleagues at the Computational Intelligence Group, Free University Amsterdam, describe the system currently on display as part of an Escher exhibition at the City Museum, the Hague, at the time of writing.

13 CHAPTER

Eons of Genetically Evolved Algorithmic Images

Steven Rooke

13.1 INTRODUCTION

In multicellular organisms, all cells except for germ-line cells (sperm and eggs) contain exactly the same DNA blueprint for their own construction. Yet those identical instructions are carried out sensing their location in the body, and they create different cell types appropriate for all the various tissues and organs needed by the adult organism.

This chapter is concerned with computer-generated images produced by a single algorithm applied equally to all pixels: the algorithm is the DNA, and the x- and y-coordinates (one unique pair per pixel) are the location in the "body," or image. The algorithms are evolved using simulated genetics from purely mathematical or programmatic building blocks that are assembled randomly into populations of individuals and later modified by mutation and mating, under user-applied aesthetic selection.

13.2 USING GP FOR ART

Richard Dawkins introduced a program in 1987, *The Blind Watchmaker,* for evolving branched stick-figure forms based on an underlying genome, through repeated cycles of mutation and selection (Dawkins 1987). In 1991 Karl Sims pioneered the technique of evolving colored raster images and animations using LISP expressions as both the genome and the runtime algorithm, adding mating of pairs of genomes to mutation as a means for genetic variation (Sims 1991). John Koza described a very general, as well as detailed, process for *genetic programming* using LISP S-expressions as both genetic code and program in 1992 (Koza 1992).

My first steps in this area were made when I constructed an image evolution system in LISP and C built around Professor Koza's genetic programming kernel, in 1993. However, on an SGI Indigo of that era, the LISP overhead made image evolution about a hundred times too slow to be practical (genetic operations performed adequately, but not per-pixel image *evaluation* of the genome). I went back to Karl Sims's 1991 paper and built a new system in C from the ground up, tailored for image evolution on SGI hardware. (Here I will be concerned only with two-dimensional images, where the same evolved program is run afresh at each pixel to produce the phenotype, or image.)

As data structures, the genomes are trees with sizes limited only by computer memory and the user's patience awaiting lengthy image evaluations. Leaves in the tree (Koza's *terminals*) are x- or y-coordinates, constants, or random numbers, while nonleaf nodes are functions of fixed arity (number of arguments). A bushy tree may contain numerous instances of coordinates x and y, which all take on the same value for a given pixel evaluation. Hence, a pixel's Cartesian or polar coordinates themselves become the primary raw materials that eventually yield its color. Functions include pretty much anything in the math library (protected against divide-by-zero etc., following Koza), plus user-written procedures. The root of the tree is the single-valued numeric result, which is then mapped through a variable-length lookup table to produce the pixel's color.

13.2.1 Genetic Variation

In the beginning, you have to start with genomes built entirely at random from the primordial soup of functions and terminals. Progress is very slow compared to later because innovations have a cumulative effect, and it takes time to build up a pool of useful innovations. As in biological evolution, variation at the genetic level is entirely random and undirected; only after multiple generations of differential reproduction and selection can "adaptive" changes be seen in the adult phenotype.

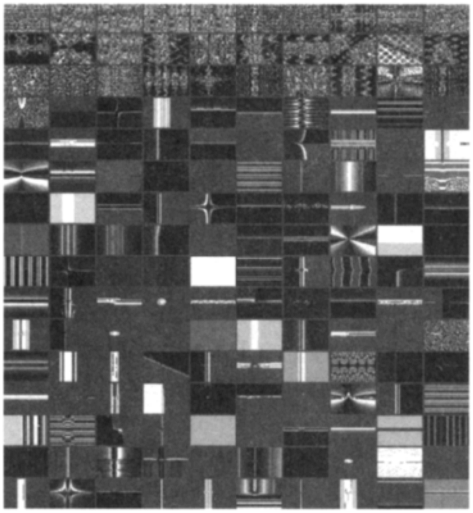
Mutation is the simplest way to generate modifications of an existing form. Say you have a population of 100 individuals in each generation. One way to get to the next generation is to select just one individual and apply 100 minor genetic mutations: Most "daughter" individuals bear some resemblance to their single "parent." At the level of functions and terminals, a point mutation means randomly selecting a particular node in the tree and replacing its function or terminal with a different one of the same arity. In image evolution, where there are hundreds of different functions, such mutation can introduce brand-new properties into a population's genome, potentially enabling large jumps in the "fitness landscape." Slight mutations of floating-point constants produce much smaller jumps.

In my experience, when the population is larger than a dozen or so, say, 100 to 200 individuals, progress is more rapid when there is a significant amount of mating relative to mutation. Unlike mutation, mating in the form of tree-structured sexual crossover (Koza's swap subtrees) can occasionally merge the portions of each parent's genome that produced useful phenotypic structures, creating in the daughter organism a whole greater than the sum of the parts. Such crossover appears most effective when repeated a random number of times with different swap points for a given pairing of individuals.

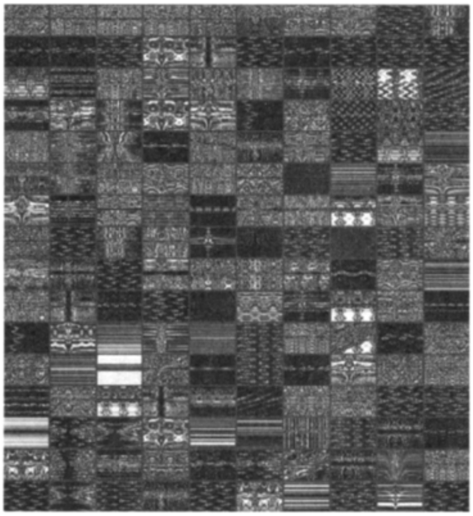
Because the populations are large and I can type much faster than I can use the mouse, I limit individual aesthetic fitness scores to a single keystroke, 0–9. All individuals start with a score of zero, so fitness assignment is a matter of quickly typing single digits or arrowing past groups of zero-valued individuals. Such coarse fitness quantization is not a problem in large populations because normalized fitness and fitness-proportionate selection of individuals are used at breeding time, following Koza. Finer-grained fitness scores, with less probabilistic mating selection, would allow the user to choose specific pairs of individuals for mating, but in my experience populations produce greater diversity when mating is somewhat more random. This is because significant sections of a genome are frequently unexpressed in an ancestor (e.g., canceling out), whereas some modification of such a section causes it to become expressed in a descendant. If you only attempt to breed the best and the brightest, you soon find yourself stuck in a genetic rut from which there is no escape.

Figure 13.1 shows a starting population built from simple arithmetic and trigonometric functions in which 31 individuals saved from previous runs have been preseeded. Figure 13.2 is the population 25 generations later and contains an image, "Afman," I decided to save, shown in Figure 13.3.

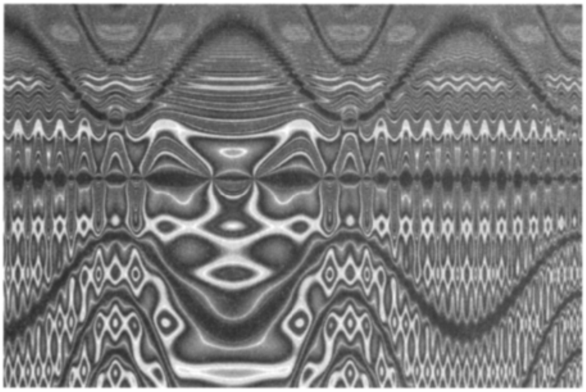
Figure 13.4 shows one of Afman's two parents. Figure 13.5 shows the genealogical tree of Afman's ancestry backward 14 generations, illustrating the various methods of reproduction: mating of two parents, mutation from a single parent, and parthenogenesis (a single parent whose genome has undergone swapping of subtrees with itself). In these runs of fixed population size,



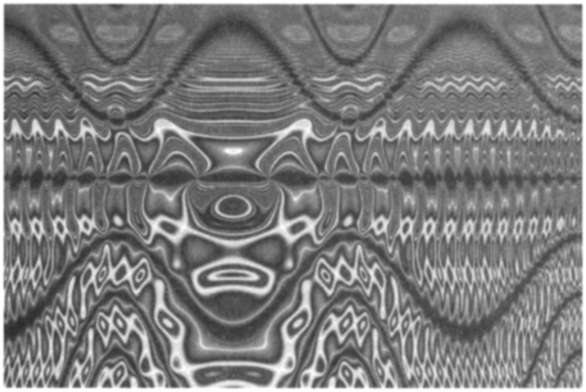
13.1 FIGURE Zeroth population with 31 seeded and 129 random individuals.



13.2 FIGURE 25th generation. Afman (Figure 13.3) is row 13, column 2.



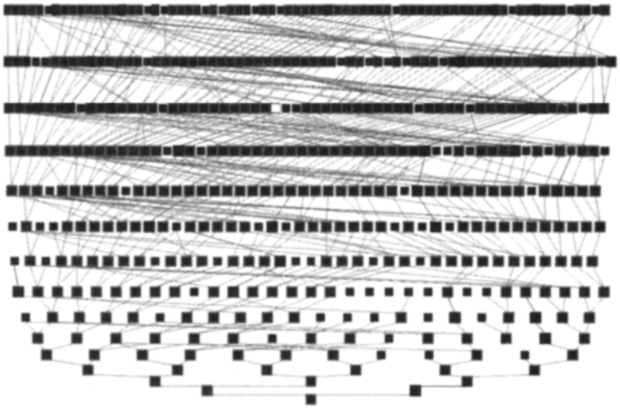
13.3 FIGURE Afman.



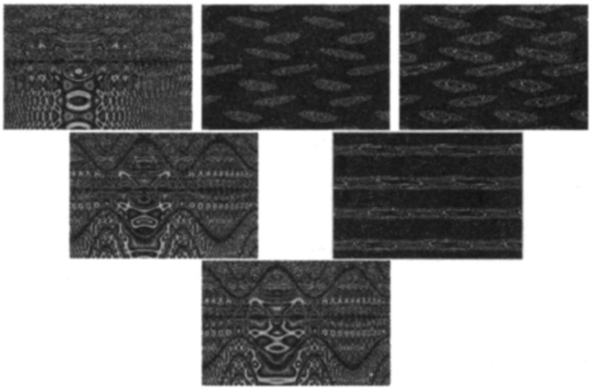
13.4 FIGURE One of Afman's two parents.

inbreeding is of course unavoidable and appears rampant in earlier generations (all those crossed lines between pairs of earlier generations).

The short genealogical tree in Figure 13.6 shows that Afman was produced by the mating of two parents and provides an unusual chance to examine a relatively simple genome to see the effect of mating. A "*y*" in one of Afman's parents was swapped with a "cos(*x*)" from the other (boxed in Figures 13.7 and 13.8), turning a somewhat repulsive "face" into a more pleasing one.



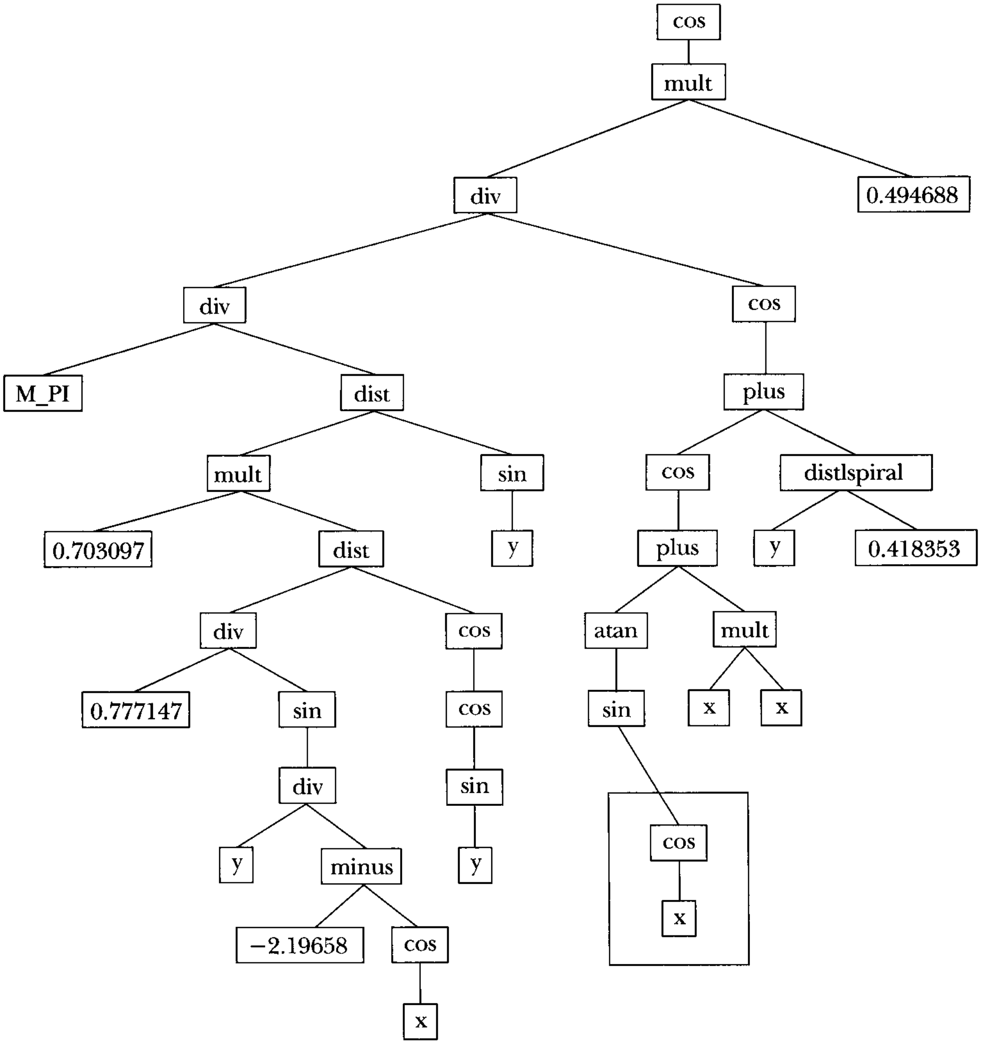
13.5 FIGURE Fourteen generations of Afman's ancestors.



13.6 FIGURE Afman's two parents and three grandparents. There is only one right-lineal grandparent, as it is reproduced by parthenogenesis.

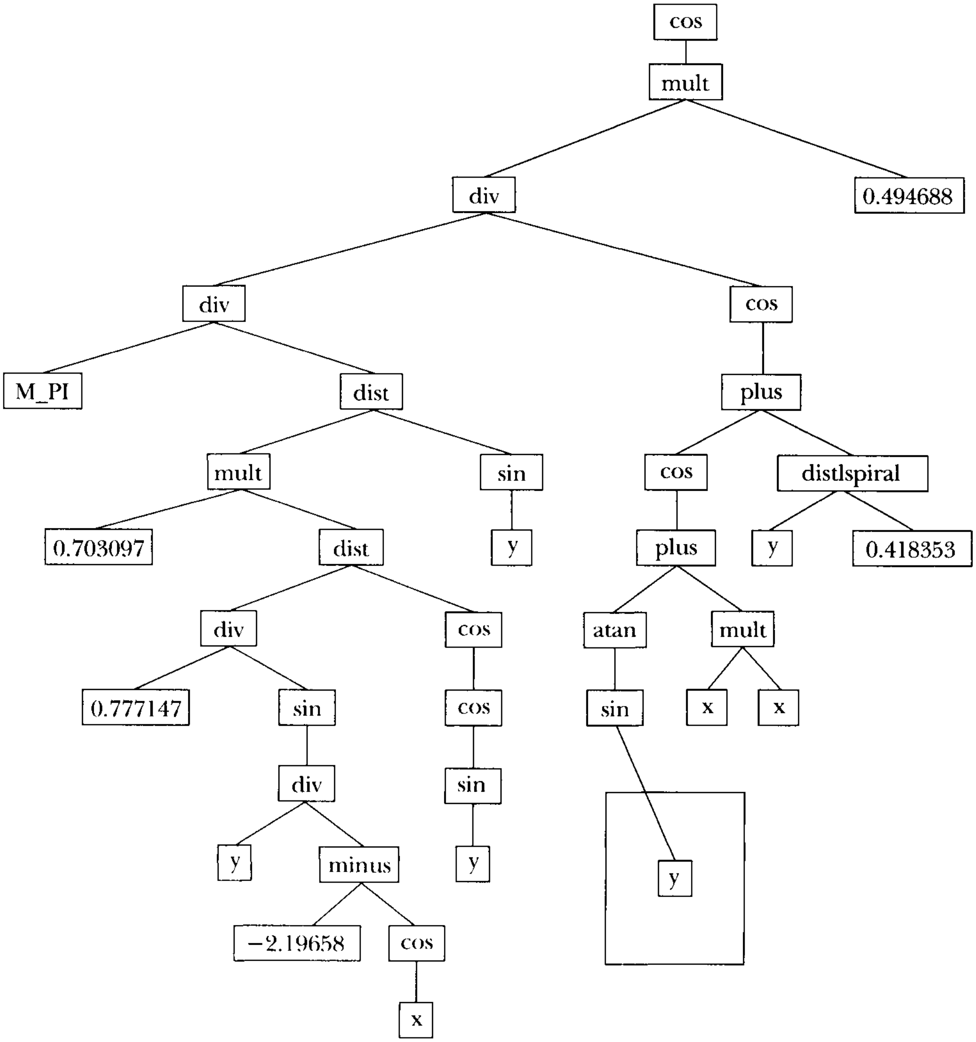
13.2.2 Genetic Library

As a practical matter, if you are to reap the rewards of cumulative genetic effects in selectively bred computer graphic images, you must have both considerable time and an extensive genetic library of all forms previously evolved (preserved in digital amber, as it were). Hence after the initial slow, rough, all-random starts,


cos
mult
div
0.494688
div
cos
M_PI
dist
plus
mult
sin
cos
distlspiral
0.703097
dist
y
plus
y
0.418353
div
cos
atan
mult
0.777147
sin
cos
sin
x
x
div
sin
y
minus
y
cos
x
−2.19658
cos
x

13.7
FIGURE

Afman's genome. The "cos(*x*)" in the box represents the only difference from his parent in Figure 13.4 (which has a "*y*").


cos
mult
div
0.494688
div
cos
M_PI
dist
plus
mult
sin
cos
distlspiral
0.703097
dist
y
plus
y
0.418353
div
cos
atan
mult
0.777147
sin
cos
sin
x
x
div
sin
y
y
minus
y
−2.19658
cos
x

13.8 FIGURE Afman's father's genome (image in Figure 13.4).

you can pick and choose a seed population from any number of promising preserved individuals and begin selectively breeding in earnest.

I store the genome in every image header in *inline* notation, which is also an executable statement in C. For example, for Afman:

```
cos(mult(div(div(M_PI,dist(mult(0.703097,dist(div(0.777147,sin(div
  (y,minus(-2.19658,cos(x))))),cos(cos(sin(y))))),sin(y))),cos(plus
  (cos(plus(atan(sin(cos(x))),mult(x,x))),distlspiral
  (y,0.418353))))),0.494688))
```

That way you can scan a bunch of images for suitability as seeds for a new run, and with a keystroke copy the genome into the seed file.

There is a trick for quickly building up a library representing an entirely separate genetic lineage (for subsequent mating with existing lineages). You can set up a random search mode, with 400 or more very small individuals in each population, and program a single key to create a new random population in background, while doing other things on the computer. As soon as the zeroth generation is done, you can see at a glance if there are any individuals worth saving, and click to save their genomes in a seed file for the new lineage. The fraction of useful individuals is very small, one or fewer per completely random population, but the process can be relatively fast. Once you build up a seed population of 100 or 200 individuals, then you can begin normal multigeneration image breeding, still using only the new lineage. Only after several runs is it wise to begin interbreeding with preexisting lineages.

13.2.3 Functions and Node Internals

The examples above included only simple arithmetic and trigonometric functions lacking initial conditions. An interesting class of functions to include in the node pool is that of iterative functions in the complex plane. So, grab any of Clifford Pickover's books (Pickover 1990, 1991) from the bookshelf, find interesting fractal algorithms, write them as compatible procedures and enter them into the function/arity table, and let the genetics worry about assembling them along with all the other (nonfractal and fractal) functions.

Say the genetics inserts a particular Mandelbrot set as a node somewhere in a bushy tree. The function expects two arguments: mandel(cReal, cImag), treating them as real and imaginary coordinates in the complex plane. If the genome just happened to supply the pixel coordinates (x,y), and mandel() were the root node, you would get the familiar Mset. But chances are that cReal and cImag are themselves the results of whole branches of functions, with many instances of coordinates *x,y* scattered out among the leaves. Enter the iteration loop, orbit around for a while, and finally escape with some measure of distance to the Mset attractor, such as the number of iterations.

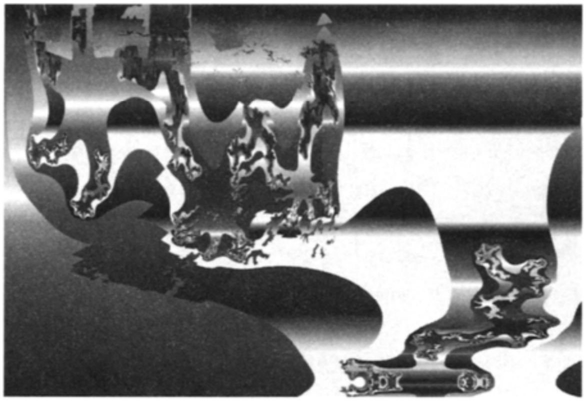
Return that number as a floating-point value, and you're back inside the genome tree, where it may feed into a geometric function [e.g., as the "*x*" in *distance* (*x,y*)] or into another dynamic function, perhaps zImag of a cubic Pickover biomorph function. Unsnarling what is actually going on in most of these genetically evolved programs is hopeless; for example, see Figures 13.9 and 13.10. Fortunately, as pointed out by Sims (1991), you don't have to: you do all your work in phenotypic space simply looking at images.

An aside on the genetic data structure: By 1994 Karl Sims had switched from trees to directed graphs (Sims 1994), noting the vastly greater capabilites of digraphs over trees. Unlike a tree, a digraph can represent a circuit, such as those evolved in Sims's creatures' supersets of neural network brains. Though circuits are not required in simple image evolution, one can imagine that a digraph genome evolved to generate an image might make profitable use of embedded circuits. After first appearing, such an image might shimmer and change with time, perhaps discovering new dynamic basins of attraction. My early experiments with digraph genomes were encouraging, except that image evaluation was prohibitively slow. This may be more of a problem in my implementation than an inherent characteristic of directed graphs, and I hope someday to return to them.

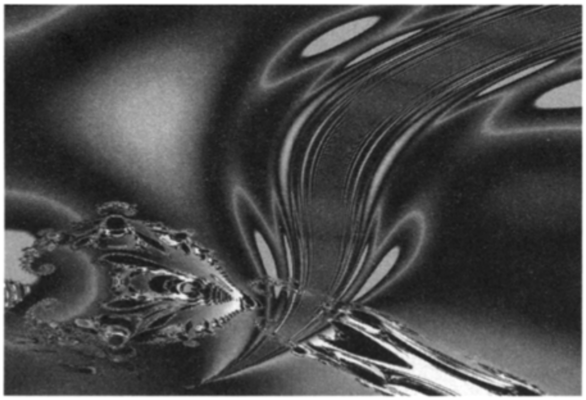
13.2.4 A Typical Run

There are dozens of parameters that govern population makeup, breeding, initial states of functions, and so on (though mostly you start by copying some recent parameter set and make only a few changes). The breeding ratios are important (cloning : mating : mutation : new-random : reseeding), and are determined by what you are trying to accomplish. It is generally a good idea to clone a number of the fittest individuals from the previous generation directly into the daughter generation, to keep their genes around intact through several generations. If you seeded the run with many preserved genomes each having a long history, you want plenty of mating. If you haven't seen anything new in a while, increase the mutation ratio—unlike mating, mutation can break out into parts of the function pool not yet incorporated into someone's genome.

Next I select a colormap. Colormaps may be changed during a run, but I find that within a particular run, image evolution "works best" with a single colormap. Likewise, after a run has been terminated, a majority of the images



13.9 FIGURE *Darwin meets Dali.* Genome includes multiple Pickover biomorphs.



13.10 FIGURE *The Crossing.* Two quaternion Julia sets surrounded by 52 nonfractal geometric functions.

saved for further work tend to look best with the same or fairly similar colormaps as the one under which breeding was conducted.

Then the cycles of selection and breeding begin, which consist mainly of typing a numeric fitness score over each nonzero individual, hitting the Breed key, and waiting. Because the images are so small at this stage, one frequently expands a number of them into larger secondary windows before deciding what score to assign. I am on my third generation of hardware since 1993, but most runs seem to take two or three days regardless; each time the hardware gets faster, I end up expanding the genome size or complexity of runtime functions. With a typical breeding ratio of 20% clones, 40% mating, and 40% mutation,and a population size of 100, it seems you start losing diversity after 40 or 50 generations,and it is time to terminate the run (though that could be just getting tired).

When you later examine an individual's genealogical tree, you can usually recall having seen the parents, grandparents, and a few striking ancestors. However, back several generations you start seeing individuals you would never have deliberately selected for breeding; the image may be all one color or have a simple line or bar in it. Frequently such images contain "unexpressed" genes; perhaps an entire branch of the genome tree is being multiplied by zero. Downstream genetic shuffling then plucks the unexpressed branch and inserts it somewhere else. Hence the value of "keeping runts in the genome"—they may not look like much now, but their genes become indispensable later. As long as a population maintains sufficient diversity, fitness-proportionate selection takes care of incorporating such marginal individuals in the generation during which they are bred. But if all images start looking much the same, you're best off starting over because experience shows it is very difficult even with mutation to break out of that kind of rut, at least with the genome sizes and complexities I tend to employ.

During the run, I usually save 20 or 30 individuals for the postevolution phase, so upon termination the first order of business is to cull that number as yet another level of selection. "Saving" an image actually means having the program write a C source file, compiling it, and writing a medium-sized image (the header of which contains a copy of the genome). As a C expression the genome can be optimized inline, so subsequent image evaluation is faster than traversing the tree recursively in an interpreter as during evolution.

Next I reframe each image, cropping, stretching, and sometimes resetting the corners as the four points of a trapezoid so as to exclude near-edge features in the completed rectangular image. When this is complete for all images, I pull in a dozen or two colormaps from other runs that might show promise. Each cropped image now generates a number of versions in different colormaps, of which one (most often the original or a close relative) is chosen for the final image. I store colormaps as vector tie points in red, green, and blue so that in the final step I can greatly increase the total number of rgb intermediates through interpolation, for a more continuous color gamut.

Algorithmic images are great when it comes to making large prints because they can be regenerated at any resolution (there's no such thing as too much resolution!). A typical large IRIS print fills over 400 MB on a CD-ROM as a color-separated CMYK image. Even at 300 dpi, it is best to supersample pixels randomly on a 4 × 4 grid for prints or 35mm slide film recorder. In principle, you could write a driver that would spin out a continuous high-resolution image onto roll media containing trillions of pixels.

13.3 HORIZON LINES AND FANTASY LANDSCAPES

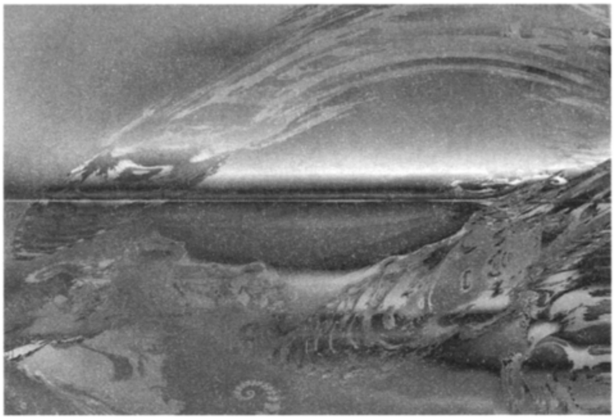
You would think if you pushed long and hard enough, with large and rich function sets, eventually you would be able to breed just about any image (see Kevin Kelley's conjecture about creating the Mona Lisa in Kelly 1994). In 1996 I returned from a trip during which I received a lot of criticism about the images being too "flat" and textury. I determined to see if I could evolve more of a landscape look, scouring the genetic library, heating up mutation, undertaking extensive breeding programs. It took about two weeks (for example, Figures 13.11 and 13.12), and I couldn't see any obvious difference in the underlying genomes, just some subtle rearrangement—but once there, the "mother of all landscapes" produced numerous offspring.

13.4 GENETIC FRACTALS

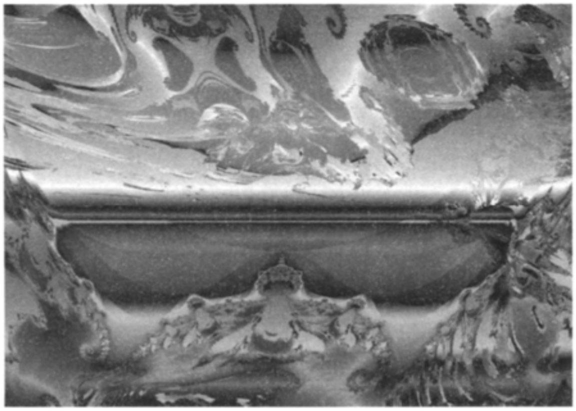
In genetic cross dissolves (see Section 13.5), the fractal spiral arms and other features in *Through Caverns Measureless* (Figure 13.13) wander across the background. Even regions of the "background" that appear not to be occluded by anything fractal, change location and magnification as the iteration count for the Julia set is modified, demonstrating that image genomes constructed out of complex mixtures of dynamic and nondynamic functions result in more than simple juxtaposition of the various forms.

Figure 13.14 appears to show structures internal to the Mandelbrot set; however, that is an artifact of nested superposition, which is why I do not call it a "genetic fractal." The 59 functions in its genome include the nine fractal ones listed in the caption. Their incorporation into a single program statement (genome) somehow yields the image. This kind of image is about as far as one can go where each embedded fractal function is a hard-coded "black box." To go further, the power of genetics and selection must open the black box to gain leverage down inside the iteration loops in the complex plane, the genetic fractals discussed below.

Color Plate 5, *Skaters,* shows that even black-box fractals in nested expressions can produce lovely images. It consists of the following genome, too large to show as a tree:

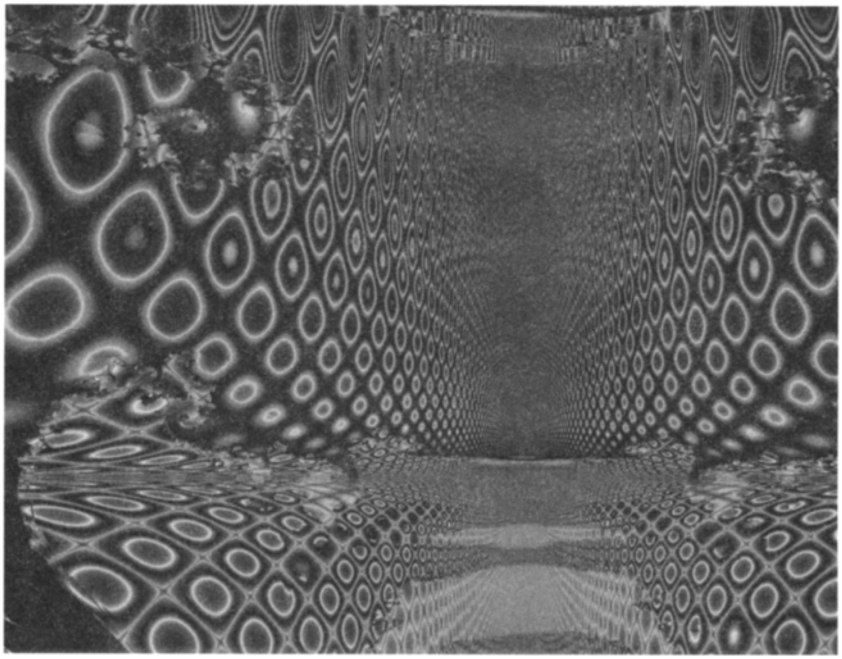


13.11 FIGURE *Primordial Yearning.*

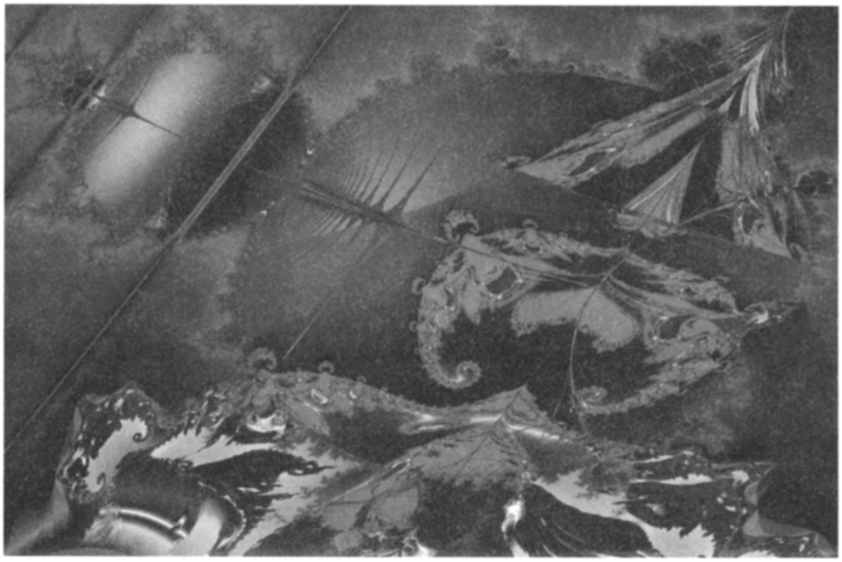


13.12 FIGURE A descendant.

```
sin(pdiv(pdiv(pminus(sin(mandelstalk(x,y)),pqj4da2013(x,pminus(sin(p
mult(atan(pdist(pln(0.290498),M_PI)),pminus(pminus(pminus(sin(y),psq
rt(x)),pln(pinvert(pmandel(pmandel(pqj4da2301(psqrt(pbmsinzpez(atan(
pmandel(pplus(psqrt(pminus(pinvert(atan(pqj4da0213(y,psqrt(pmult(pmi
nus(y,pminus(y,pminus(M_PI,x))),y)))))),x)),y),M_PI)),M_E)),x),M_PI),
0.412728)))),M_PI))),y))),pthetlspiral(pplus(pln(0.804698),pdistlspi
ral(pdiv(cos(y),sin(pmult(psqrt(sin(psqrt(pmult(pdiv(cos(x),0.022307
),M_PI)))),pmult(M_PI,y))))),y)),y)),y))
```



13.13 FIGURE *Through Caverns Measureless.* Changing the iteration count in the 4D quaternion Julia set (the only fractal among 37 functions in the genome) changes the entire image, not just the parts that look fractal here.



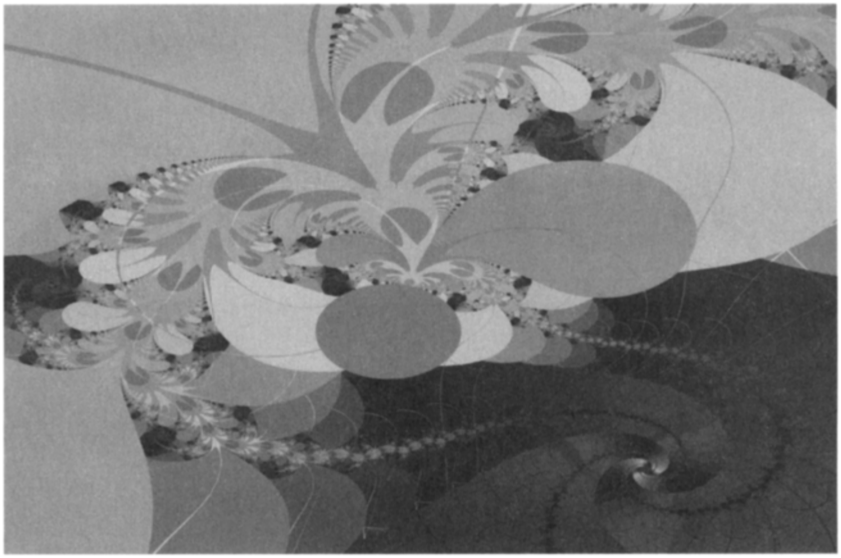
13.14 FIGURE This is *not* a "genetic fractal." Its genome contains four Msets, three different slices of 4D quaternion Jsets, two Pickover biomorphs, and numerous nonfractal functions, but only as "black boxes."

This includes four Mandelbrot sets (one with Pickover stalks), three 4D quaternion Julia sets computed along different 2D slices, and one Pickover biomorph ($z \to \sin(z) + e^z$), embedded among 22 arithmetic and 11 trigonometric functions, 12 logarithmic and distance functions, seven occurrences of π, two other constants, and 6 x- and 13 y-coordinates. I still find it somewhat mind-blowing that *every* pixel is computed through *all* of these 53 functions and 28 terminals in a genome like this.

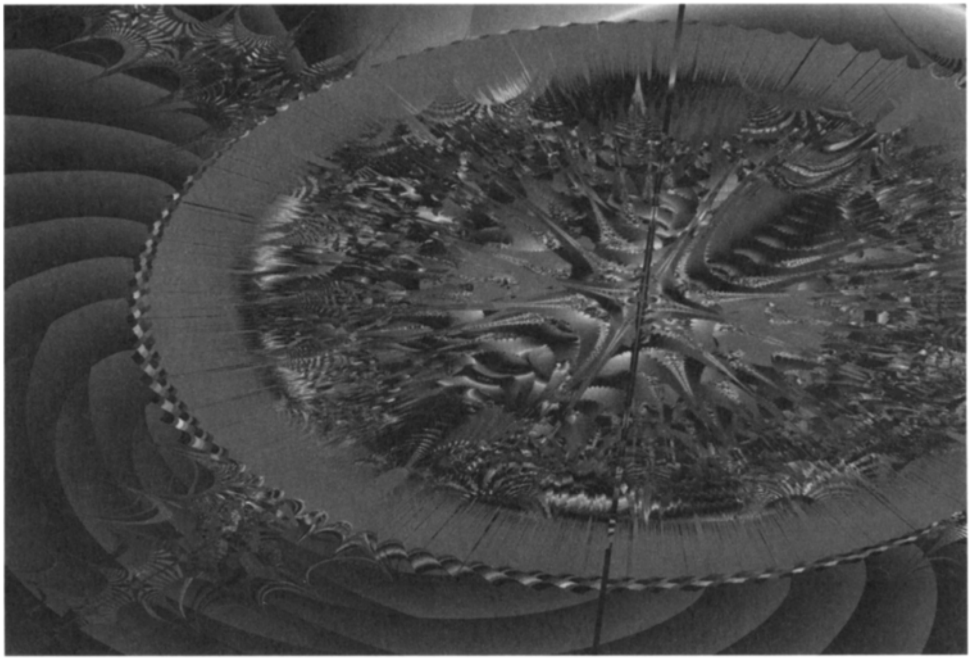
In late 1997 I lurked on the Fractint mailing list for two weeks and was amazed by the creativity of the group, people posting "fractals of the day," some of which looked like nothing I had seen (the Fractint home page is *http://spanky.triumf.ca/www/fractint/fractint.html*). At the time, Fractint was only available on PC and Macintosh platforms, so I did not download the code. However, one of their Web documents described a parser that allowed users to construct their own new fractal equations. For example, the Mandelbrot set is $z \to z^2 + c$. What would an image look like if the equation were $z \to \sin(z^3 - 1/z)$? The Fractint designers allowed users to type in any such expression, then wait while the interpreter tried it out. Some minutes or hours later, the user would see the result and perhaps go back to tweak some of the input parameters. People were coming up with some very interesting images, but the trial-and-error nature of the process appeared quite slow to me.

I reasoned that I could write a genetic generator similar to the Fractint parser and let the power of genetics and selection figure out how to create the interesting new equations while viewing only the resulting phenotypes. Just as the sun was setting on the winter solstice 1997, my first *genetic fractal* saw the light of day (Figure 13.15). The machinations required to give the genetics leverage all the way down inside the iteration loops in the complex plane required a minor new genetic minilanguage, rather than an extension of the older one. I ended up nesting the new genetic fractal expressions and iteration control schemes inside a single node—any node or nodes—of the outer genome. So now, if the outer (old) genome consisted of "genlIter07(x,y)," the new, nested genome might be $z \to e^{c/z}$, where c is x, and z is y, as in figure 13.15.

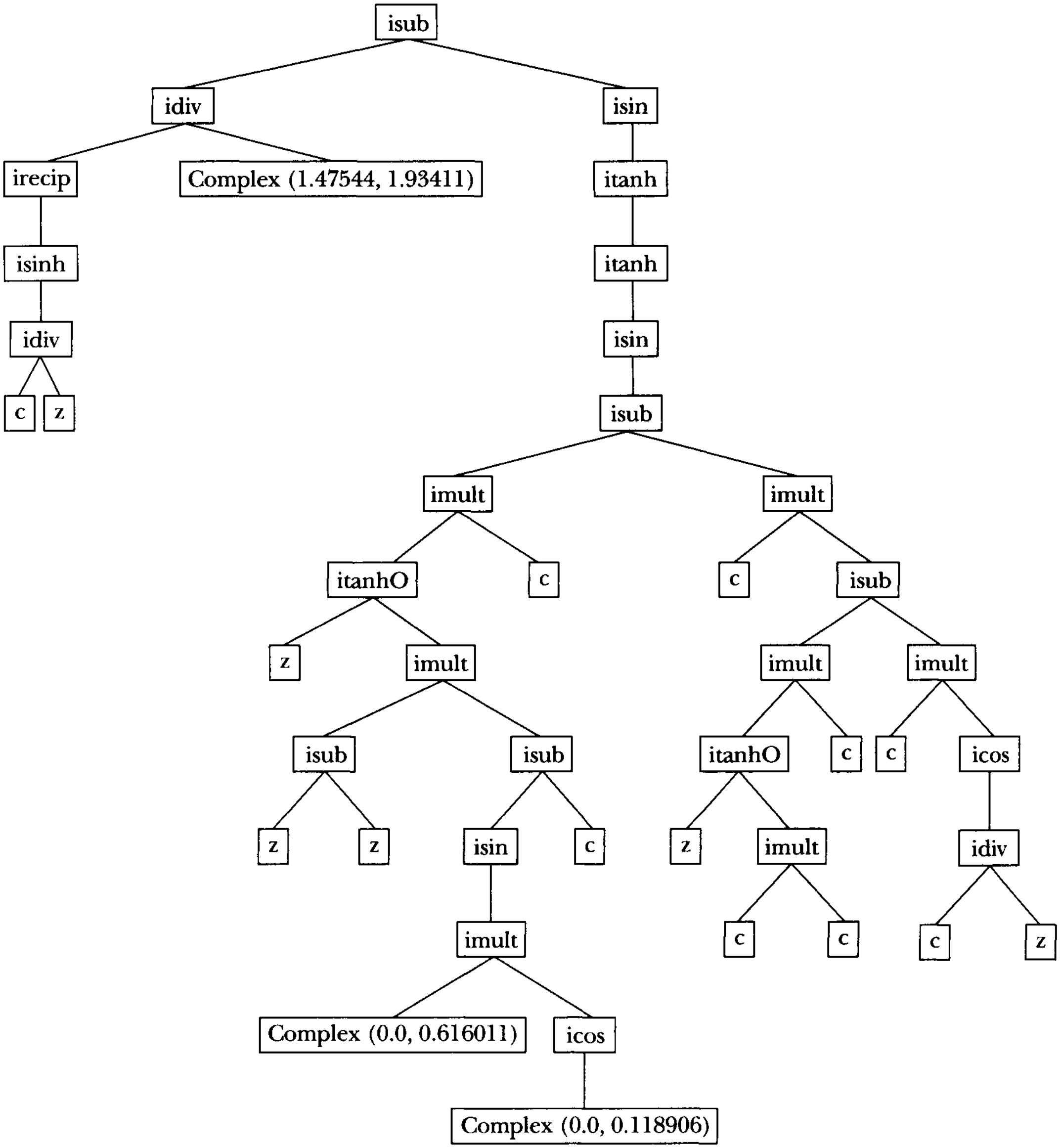
But that inner, nested expression could now become arbitrarily complex, trying out a number of different initializations, iteration-exit strategies, distance measures from attractors, and so on. Figure 13.16 shows an image whose genome contains the iteration expression shown in tree form in Figure 13.17. Figure 13.18 shows various other genetic fractals.



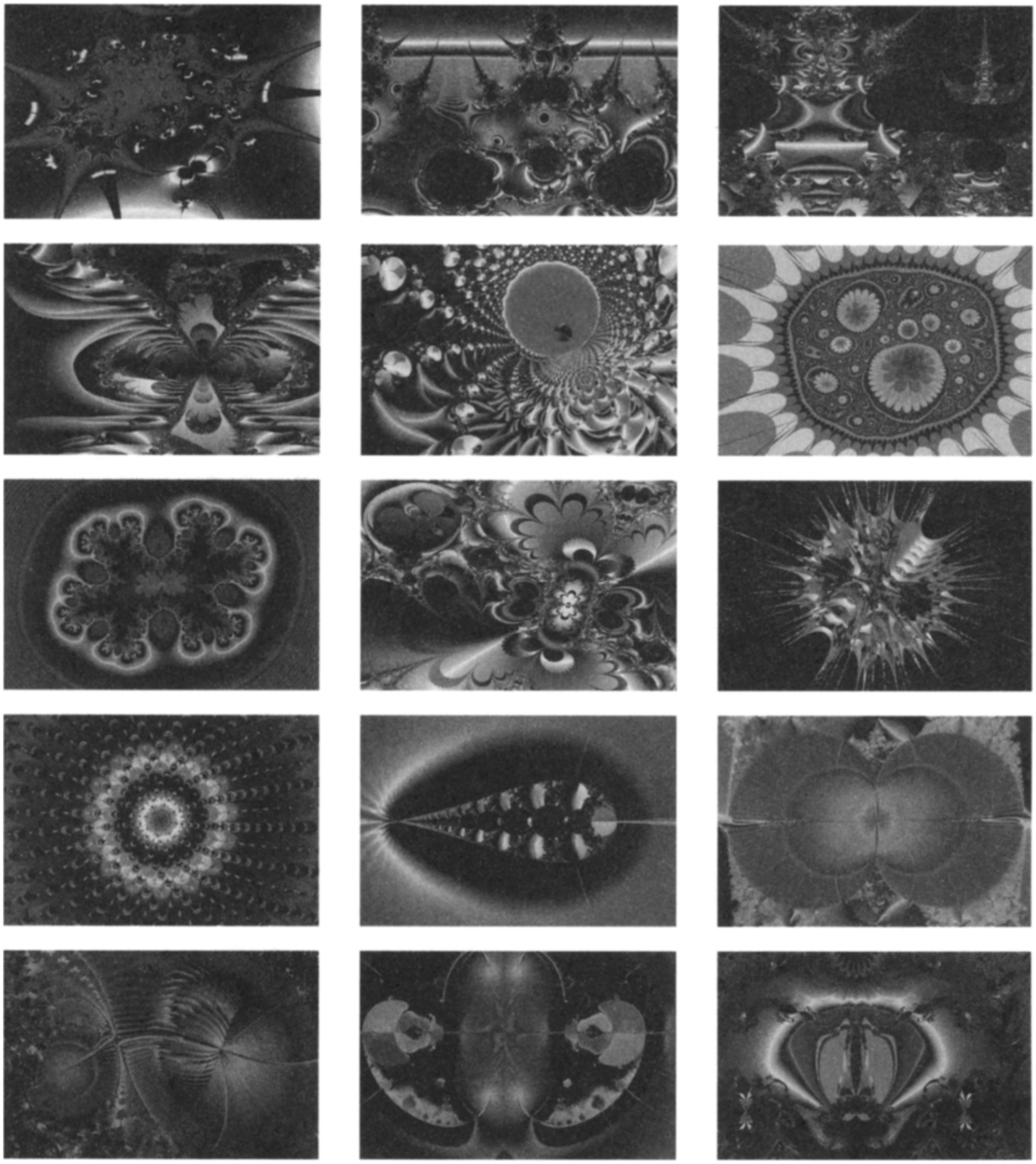
13.15 FIGURE *Winter Solstice 1997* genetic fractal, $z \to e^{c/z}$.



13.16 FIGURE A genetic fractal. The iteration expression is shown in Figure 13.17.


isub
idiv
isin
irecip
Complex (1.47544, 1.93411)
itanh
isinh
itanh
idiv
isin
c
z
isub
imult
imult
itanhO
c
c
isub
z
imult
imult
imult
isub
isub
itanhO
c
c
icos
z
z
isin
c
z
imult
idiv
imult
c
c
c
z
Complex (0.0, 0.616011)
icos
Complex (0.0, 0.118906)

13.17 FIGURE Inner (nested) fractal iteration portion of genome for Figure 13.16.



13.18. FIGURE Various genetic fractals in the complex plane.

13.4.1 Second-Order Subtleties of Orbit Trajectories during Iteration in the Complex Plane

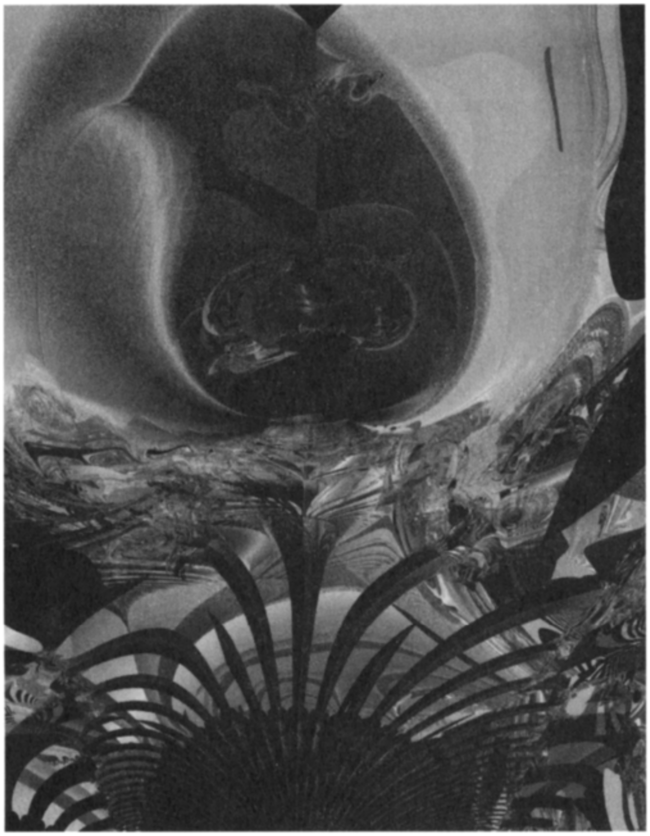
In the simplest rendering of the familiar Mandelbrot Set, $z \rightarrow z^2 + c$, at each step during iteration the new z is calculated, representing a new point in the complex plane. Usually the distance of the new z from the origin is compared against a maximum distance and if exceeded, iteration is terminated and the total number of iteration steps is returned as a measure of the distance of the initial z from the Mset attractor.

The literature is full of more sophisticated quantities than a simple iteration count. Armed with a genetic search engine, I decided to try some brand-new ones. Each new z forms a line segment back to the previous z, and so on, forming angles between each pair of line segments. Store the difference $\theta_{current} - \theta_{last}$ in an array, one element per iteration step. Let the genetics try out various ways of deciding how to initialize, when to escape the iteration loop, and so on, and send that array of angles through a statistical moment routine upon escape. Figure 13.19 is an image of the *standard deviation* at each pixel of those changes in angle from one iteration to the next, for the equation $z \rightarrow \tanh(inewtl(z, iconj(z))^{1/z})^c$. It is surprising to see this coherent an apparent spatial structure emerging from as derivative a measure as the variance of a change in angle from one iteration step to the next. (Color Plate 6 shows a different version in color.)

13.5 THE GENETIC CROSS DISSOLVE

Karl Sims introduced the *genetic cross dissolve* in 1991 (Sims 1991). Picture genomes A and B, which have the same function for the root node, and perhaps for some distance along the branches, but which then diverge after one or many points of last contact. A and B are genetically related, more closely (recently) if there are few differences, more distantly (usually farther back in time) if the differences are many. Make a list in your mind of each point of divergence between the two genomes (it is done differently in a program, but I find it easier to visualize this way). Because the result of all calculations below each divergence point, as below all nodes, is a floating-point number to be fed back up the line, there is a *continuum* of images that can be evaluated between genomes A and B.

Now say you want to produce 10 seconds of video animation between A and B (that is 300 frames in NTSC video). The continuum is now sampled at 300 discrete points: The program marches along the genetic branches of both genomes

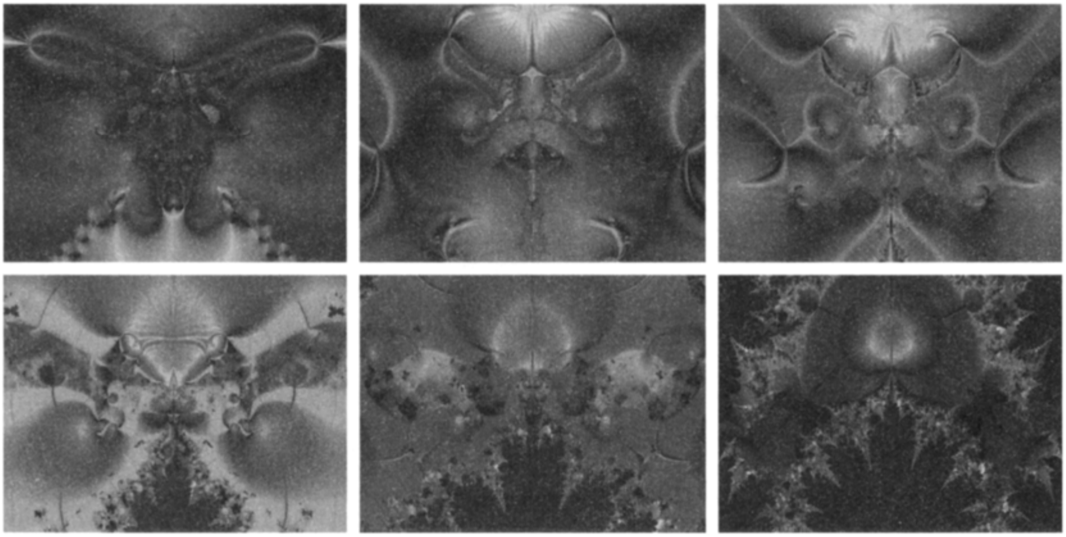


13.19
FIGURE

Hyperspace Embryo, a true genetic fractal.

until it reaches a node of divergence. At that point it then computes separate values for the now-different downstream branches of the A and B genomes. The value fed back upstream into the identical branches is simply the value interpolated for the current frame or time step (e.g., 0.333A + 0.667B for frame no. 100/300).

In a sense, this genetic cross dissolve is tracing a trajectory through a number of previously unseen genetic relatives. One can imagine plotting the genealogical trees for individuals A and B and determining that a particular intermediate gencross frame is A's great-great-grandmother on the maternal side, which happens to be one of B's eleventh-great-grandparents, but most of the time an intermediate frame will be a fractional virtual relative unreachable by normal integral genetics.



13.20 FIGURE Six frames from a genetic cross dissolve (left to right, top to bottom) between just two genetic fractal keyframes A, B.

Genetic cross dissolves frequently generate unexpected forms intermediate to the two end members. Figure 13.20 is a series of frames between just two genetic endpoints. Individual A has the iteration expression $z \leftarrow i\cosh(\mathit{isub}(\mathit{irecip}(i\cosh(\mathit{imust}(\mathit{irecip}(\mathit{isum}(\mathit{isub}(\mathit{irecip}(\mathit{isum}(z,c)),z),c)),z))),\mathit{irecip}(\mathit{isum}(z,z))))$. Individual B has the iteration expression $z \leftarrow i\cosh(\mathit{isub}(\mathit{irecip}(\mathit{isum}(z,c)),\mathit{isum}(\mathit{isub}(\mathit{irecip}(\mathit{isum}(z,c)),z),c)))$. It is not possible to guess what the intermediates will look like by simply examining the two end members. Of course, genetic cross dissolves are best experienced as animation. Usually one provides a list of previously evolved individuals, and the program employs curve-fitting such as NURBS (Non-Uniform Rational B-Splines; see Foley et al. 1996) to smooth the trajectories through genetic, coordinate, and color spaces.

13.6 WHAT IS IT?

Nothing generates more controversy than to take a bunch of these images into a roomful of experts and ask, "Is it art?" Where's the human context, the critical discourse? Certainly all such purely algorithmic images are more *found* than *created,* in the sense that the end product is a mathematical expression that exists as surely outside the universe of space and time as does any statement of abstract number theory or plane geometry. (One might quibble about anthropomorphic details like the sense of sight and color, but in principle there should be a mapping from pictures perceived by human sight into any form perceived by any kind of sentient creature that would reveal the same structural details).

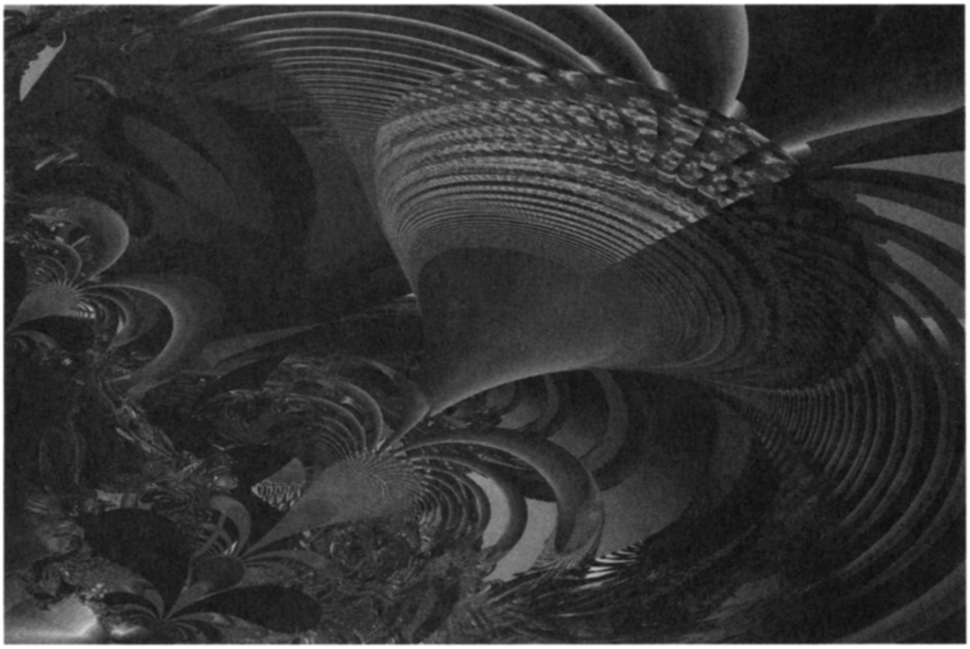
Breeding algorithmic images on a computer is more like photography than painting, in the sense that when you "snap the shutter" you get everything that was in the field of view rather than composing separate parts differentially. Like photography, the field of view is just a window into a much larger (perhaps infinite) viewscape. As with photography, one has extensive control of framing, orientation, and color (with the constraint concerning colormaps noted below). Unlike photography, however, the contents of the scene themselves are mutable, and under considerable control of the "evolographer"—not in the current, or "final" image, but in loosely guiding its breeding by assigning aesthetic fitness scores to its thousands of ancestors throughout hundreds or thousands of prior generations. Each image has a lengthy history involving innumerable choices by the user (and involving considerable randomness, as well).

Except for the genetic and selection aspects, image evolution as a process of discovery is not unlike exploring the marvels in a drop of pond or sea water with a microscope. The territory is unfamiliar at first, but gradually begins to reveal the most fantastic creatures—Ernst Haeckel's illustrations come to mind (Haekel 1974). Unlike the microscope, the computer as an instrument of algorithmic exploration has an infinite depth of field, so there is often an extensive background behind a framed "animalcule."

Evolved images are "found," but found only by means of a very intricate and lengthy process that has creative aspects. There are similarities to selective breeding of plants or animals, particularly in selecting breeding stock, and in the constant winnowing out of those less favored. Unlike selective breeding of purely existing stock, the programmer continually adds new functions and procedures, sometimes new genetic coding schemas, rapidly broadening the range of what is possible. Perhaps we will see a similar phenomenon with designer genetic engineering in the new century.

13.6.1 Constraints of Color and Form

In this chapter, all images are generated with a single floating-point value for each pixel, scaled between -1.0 and 1.0 before being processed through a colormap. Suppose a pixel in one part of the image has the value 0.42, which maps to



13.21 FIGURE Standard deviation of the vector of differences in angle from one line segment during iteration to the next, with the formula $z \leftarrow \mathit{ipow}(i \tanh)(\mathit{ipow}(\mathit{inewt1}(z,\mathit{iconj}(z)),\mathit{irecip}(z))),\mathit{ipow}(c,\mathit{irecip}(\mathit{isub}(\mathit{complex}(0,-1.26775),\mathit{isub}(z,c)))))$.

a particular shade of lavender through the colormap. *Every* pixel anywhere in the image with a value of 0.42 must then be exactly the same shade of lavender. It is the judicious construction of the colormap with its peaks and valleys and shadows, combined with the detailed structure of the algorithmic image topography, that produces the final image, often with its illusion of depth. Of course, you could always postprocess an image and manually change the colors in some regions differently than in others, but a true Algorist considers that to be "cheating." (Jean-Pierre Hébert discusses the derivation of the term *algorist* as an artist who uses algorithms on *http://www.solo.com/studio/algo-def.html.)*

I believe there are good reasons for preserving the colormap/value integrity of an image. The image is generated not by some haphazard jumbling of functions and terminals. Only mathematically consistent forms are possible, imposing a hidden kind of ordering of structural elements. Consider the image in Figure 13.21: There is an impression of three-dimensional consistency, with sweeping fanlike structures in the foreground occluding more distant forms in the background. The foreground arms are not entangled like spaghetti with background arms, and there is a sense of shapes receding into the distance. One has to remind oneself that in these images there is no explicit geometry. Each pixel is calculated independently of every other pixel using exactly the same program. The only difference from one pixel to the next is its position, the value of its (x,y) coordinate.

Surely a great painter places each dot of paint in just such a way that the whole picture holds together, rather than letting the dot fall at some random location, say, when the telephone rings, startling the artist. In some similar way each element of an algorithmic image falls into a position that "makes sense" relative to all other elements automatically because of the underlying structure. Of course not just any old evolved algorithmic image will do—it is the repeated application of aesthetic selection upon all the ancestors of an image, relative to all the possible might-have-been ancestors (as it were), that yields an evolutionary image worthy of surviving all the hurdles placed in its way, finally to become "real" and shown to others. Usually one can detect postalgorithmic tampering—something just isn't right.

Hence programmers and other nonartists like myself are able to harness a little piece of a process as old as the Universe—evolution—to produce imagery that we would never be able to conceive and make real by traditional means. I believe this capability is something new, just as the advent of photography enabled people who could see things in Nature not previously revealed in any artist's painted works to express themselves visually.

13.6.2 A Joyride for the Visual Cortex?

Often I have shown slides to an audience in the evening, to be approached the next day by people saying they saw these images in their dreams. Of course I see them in my own dreams all the time, so I ask them if the images were like still snapshots from a photo album, or something different. A number of people report seeing the images move—parts of the image move differentially with respect to others. Well, sure, we have dreams all the time where the scenery shifts against the background as we move through it.

But just possibly there is something more interesting going on here: These calculations must be quite trivial for the massively parallel visual computer that is our visual cortex. Perhaps parts of the brain are constantly performing something like the per-pixel calculations in these images as a normal, hidden, low-level stage in the process of vision. By some estimates the human visual cortex comprises roughly half of our computational processing power. Might the dreamtime brain, disconnected from active visual input, and having seen algorithmic images like these, go off on a joyride, performing similar calculations on its own in real time and generating its own algorithmic genetic cross dissolves?

13.6.3 Approaching the Organic

In my experience, each new plateau of complexity of underlying expression has resulted in visual forms that simply look more "organic." I would like to recount here an observation made by Marcos Novak at the Art and Aesthetics of Artificial Life exhibit at UCLA on July 12, 1998 (paraphrased): "As the underlying expression becomes more complex, the resulting forms appear more organic."

Novak specifically included expressions in higher dimensions as among those that produced more organic-looking results. For example, he constructs software that algorithmically generates three-dimensional architecture. When the algorithm was written in four dimensions, the three-dimensional slice architecture drawn by the program just "looks" more organic. Informationally, the algorithmic images in this chapter may be thought of as 2D windows into a higher-dimensional space determined by the many degrees of freedom available to tree-structured programs comprised of a large number of functions and terminals.

On one level, Novak's observation is self-evident: Before the first self-replicating organic molecules evolved (anywhere in the universe), all informational expressions in nature were simpler than the organic ones that would follow. If you write out the molecular expression for the most complex inorganic unit crystal there is, then contrast it with that of very simple organic molecules, the underlying expression for the organic molecules wins hands down. But on another level, Novak's assertion makes a lot of sense as a universal truth and could be highly relevant for our creations as we plunge headlong into the Information Age. Perhaps what we call "organic" is simply any phenomenon generated by processes whose symbolic expression exceeds some threshold of complexity.

13.7 CONCLUSIONS

Mine has been a rather extensive exploration of an exceedingly narrow niche of evolutionary computation, namely, that of 2D images lacking any explicit geometry, wherein each pixel computes a single value independently from exactly the same evolved program as every other pixel, and then are colormapped. Contained within this narrow exploration are many lessons from genetics and breeding (e.g., the importance of diversity). The evolutionary search strategy of variation and selection is a powerful tool for discovering rich new algorithms such as the genetic fractals, like veins of concentrated ore sparsely scattered throughout an enormous wasteland.

REFERENCES

Dawkins, R. (1987). *The Blind Watchmaker.* Norton.

Foley, J. D., A. van Dam, S. K. Feiner, J. F. Hughes, and R. L. Phillips (1996). *Computer Graphics Principles and Practice, Second Edition in C.* Addison-Wesley.

Haeckel, E. (1974). *Art Forms in Nature.* Prestel-Verlag.

Kelly, K. (1994). *Out of Control—The Rise of Neo-Biological Civilization.* Addison-Wesley.

Koza, J. (1992). *Genetic Programming—On the Programming of Computers by Means of Natural Selection.* MIT Press.

Pickover, C. A. (1990). *Computers, Pattern, Chaos and Beauty—Graphics from an Unseen World.* St. Martin's Press.

Pickover, C. A. (1991). *Computers and the Imagination—Visual Adventures Beyond the Edge.* St. Martin's Press.

Sims, K. (1991). Artificial Evolution for Computer Graphics. In *1991 ACM SIGGRAPH Proceedings, Computer Graphics 25*(4).

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. In R. A. Brooks and P. Maes (eds.), *Artificial Life IV, Proceedings of the Fourth International Workshop on the Synthesis and Simulation of Living Systems.*

This Page Intentionally Left Blank

14 CHAPTER Art, Robots, and Evolution as a Tool for Creativity

Luigi Pagliarini University of Aarhus, Denmark
Henrik Hautop Lund University of Aarhus, Denmark

14.1 INTRODUCTION

In recent years, electronics has expanded its presence to most of our daily life. No matter whether you prefer virtual or physical reality and regardless of personal likes, dislikes, and preferences, it is impossible to shut electronics out of our life (or out of the life of society). Moreover, there are so many facets to the electronic cultural revolution that one feels there is no limit to how much that could be said about it. Here, however, we will discuss one particular aspect of the revolution: the impact of the electronic revolution on artistic thought. We will try to do this without moving too far away from art, while at the same time avoiding too detailed an analysis, so as not to get lost.

We begin by examining the relationship between art and old technologies, pointing to the evolutionary, as well as the revolutionary, impact of new means of expression. Against this background we propose a definition for what we call "Alive Art," and develop a tentative profile of the performers (the "Alivers"). Alive Art is based on the principle of perpetual change, interactivity, and vulnerability. Later, we present examples showing what an Alive Artwork might look like and the way in which the performer of Alive Art might lose his/her artistic authority, in the sense of losing (more or less partially) direct control of the art piece realization. Finally, we will analyze the mechanisms and dynamics of the coming cultural revolution. The focus is on the "artistic space" where the revolution is taking place, on the interactions between the artistic act and the space in which the act takes place, on the way in which the act modifies the space, and on how the space modifies the act. To help explain, we will describe two examples of Alive Artworks (the Artificial Painter and LEGO MINDSTORMS robots), pointing out the central role of what we call the "Alive Art effect" in which we can perceive, perhaps, the loss of the unique creative role of the individual artist. Indeed, instruments like evolutionary programs and robotics might gradually participate in the artistic process, traditionally mediated by humans only.

14.2 THE SOCIAL CONTEXT OF ELECTRONICS

In this section we will discuss the possible relationship between electronics, as a technology, and art, in the sense of a specific social context. In order to identify the arts whose roots reach down into electronics we will need to understand the terrain we are discussing. For this reason, we will try to analyze the stage on which electronics plays its role.

14.2.1 Where Electronics Acts

In our opinion, there are two main spaces where electronics can interact with and perhaps challenge human beings. It is as if electronic machines have two separate identities facing inward and outward. There is one side to the machine, which exists in the physical world. This is hardware (e.g., a robot). Then there are the machine's "internals," its software (e.g., a program interface). Despite this distinction, ordinary people's reaction to the words "electronics" and "computer" is to think of a screen on a PC. It follows that this is one of the places where, socially and psychologically speaking, electronics *acts*. In some sense, a PC screen is similar to the human eye; it mirrors the soul of the electronics. What is more, recently the PC screen has acquired a new face—people can now perceive electronics in action via the Internet. Materially there has been no change, but the social sense has, however, completely changed. This previously unknown "virtual space" represented by the Internet is rapidly becoming an essential aspect of business and economics. Socially speaking, this means that it will become a pervasive aspect of our lives.

Art will have to cope with this. The Internet is a real if strange kind of space. Just as humans have always done when they have discovered new continents, they will attempt to populate it with all kinds of human artifacts. There are many "ordinary" electronic artifacts, which people do not consider to be art and which they do not see as being particularly intelligent or autonomous. These are of enormous importance—it is around such devices that some of the toughest battles will be fought in the future. These, however, lie outside the scope of our discussion here. Here we will concentrate on two main themes: autonomous robotics, and neuro-informatics—hybridized in the cyborg concept. Every day society is becoming more closely involved with these two areas of electronics. For the first time since humans began to dream about intelligent machines (e.g., Frankenstein), we have begun to see them every day on the news (Lund and Pagliarini 1999, 2000; Lund et al. 1998, 1999; Miglino, Lund, and Cardaci 1999). People now want wearable computers,[1] emotional computers (Pagliarini et al. 1996),[2] washable and edible computers! The Cyborg concept[3] inevitably follows behind and is getting more and more popular.

To summarize, electronics has two main roles in a human social context. Hardware brings electronics into the physical world; software transports the physical world into electronics. Hardware devices are physical objects in the real world, subject to physical rules; software, on the other hand, tends to propose itself as a system, governed by its own rules. These are the physical spaces and the social context where electronics acts, but what kind of relationship can we create with these spaces?

14.2.2 How Technology Influences Art (the World)

To ask how electronics influences art, and consequentially the world, might sound like a hard question. The history of human ideas and art might help us in understanding it. For example, when photography spread into human societies, art (in general) and painting (in particular) changed abruptly. Painting lost its role as a tool for imitating reality and began to explore new meanings of vision. Then photography began to evolve too—and after initial rejection by conservative critics and artists, it was accepted as a new form of art. This changed the "common sense" view of art and the rituals of perceiving art. Walter Benjamin has explained how photography, and cinema and music recording, overthrew principles that had been felt to be indispensable for art—in particular the uniqueness of original works (Benjamin 1995). In the same way, painting was influenced by photography and cinema, investigating and finding new ways of depicting movement, as in Cubism, and inventing a new vocabulary of the form, as in abstract art. This would suggest that technologies have the power to change the values of people—even the values of people who do not use them. Will electronic art influence the world in the some way? What does electronics offer us that photography, cinema, and painting cannot? Could it be interactivity? Or

---

[1] See *http://lcs.www.media.mit.edu/projects/wearables* and *http://www.wearcomp.org.*

[2] Also see *http://gracco.irmkant.rm.cnr.it/luigi/lupa_face.html.*

[3] See *http://www.stelarc.va.com.au.*

"changeability"? We think both factors count. Later, we will return to these issues.

14.2.3 How Technology Gets Feedback (from Art and the World)

Feedback is not a one-way process. There are ways in which art gets "revenge" on technologies and science. Technology allows human beings to manipulate their habitat. Art gives human meaning to this process. Pythagoras, for example, discovered the relationship between tone and the length of a string but never thought of playing the instrument he had discovered. Composers, on the other hand, continually re-discover and "explain" the magic of that same instrument. Thus, when art explores the extreme possibilities of technological products, it creates requirements for new technology. Will electronics and digital arts follow this same path? Will they be influenced by art movements? We believe this has already happened. We strongly argue that electronic works of art (Benjamin 1995; Lund, Pagliarini, and Miglino 1995a, 1995b)[4] would never have been accepted or recognized without the influence of forerunners using traditional techniques as in so-called contemporary classical music (e.g., Berio or Stockhausen) or in movements such as Futurism. Later on in this chapter, we will present evidence of this tendency. First, however, let us retrace our steps and try to figure out what we mean when we talk about an "electronic artist."

14.3 WHAT ARTIST?

Western societies usually segregate artists and scientists into two different mental spaces. Today, however, novel electronics-based technologies are giving birth to a new kind of artist who is closer to technological and scientific knowledge—artists who share the approach of a Leonardo da Vinci, who recognized no separation between art and science. To understand the need for this kind of science-oriented artist we have to look more closely at what artists actually do.

14.3.1 Two Different Concepts or Aspects of the Artist

There are many possible definitions of what it means to be an artist. Among these we can identify two main conceptions: the concept of the "immaterial" artist and that of the "material" artist. This makes it possible to distinguish, if not two different kinds of artist, at least two different categories of "artistic act." The former is based on the abstract idea, the concept, that lies behind a work of art; the latter is centered on the phenomenon: the material translation of the concept into the physical world. To become real art, an artifact requires both. Better, the first category of act, artist conception, refers to a mental process, state, or attitude that leads to the production of ideas. It has to do with language and the sense of a work of art. This is what we mean by "immaterial" art: The way in which it is produced (in this case via linguistic revolution or evolution) is the same for any kind of art. The second kind of art action, on the other hand, is much more closely related to the workings of mind, in the modern sense of a body and brain functioning as a whole. The idea of a "material" artist, in our sense, has much to do with body action (e.g., the movements of a dancer using the peripheral part of the nervous system) and with the technology the artist might use. In this context the search for new "tools" is a key part of the artistic process in which a number of artists play a pioneering role. This work is very similar to the work performed by scientists. Let us clarify with two examples.

---

[4] Also see *http://gracco.irmkant.rm.cnr.it/luigi/alg_art.htm.*

14.3.2 Art and Human Language: The "Immaterial" Artist

Art critics identify conceptual arts as these arts in which there is no transformation of matter. This corresponds to an idea of society where the relationship between the things is more important than the things themselves. The aim is to invent expressive codes, processes, and systems that will, in turn, produce aesthetic matter. We are talking of artists like Marcel Duchamp's sons/daughters. Another example might be Sergio Lombardo (see Color Plate 7) with his emphasis on the ways and means of artistic creation (Lombardo 1983). Lombardo's methodology involves sciences like stochastic mathematics and psychology. The resulting aesthetic, while a product of these disciplines, is nonetheless unpredictable. Artists like Duchamp and Lombardo can be called artists because of the way they changed the use of symbols, language, and ideas in sculpture and painting, respectively. It would not matter at all if their works had no aesthetic content (although we would miss it); the revolution in language is enough.

14.3.3 Art and Human Technique: The "Material" Artist

As we said before, part of the artistic approach to the world, besides being related to the physical skill of the artist, is a very scientific one—being closely related to the techniques or technologies the artist uses. In a way, this aspect of the art can be easily abstracted from the language and the meanings that art usually brings with it. One example might be Simone Martini, a renaissance artist. His famous "blue" color was the fruit of a deep chemical knowledge of pigments. That blue was at the time a unique aesthetic result, which many of his contemporaries tried to reach in vain (and which brought him celebrity). No doubt, it had something to do with painting and the aesthetics of painting. At the same time, however, it was a purely scientific discovery. The same can be said of Bach, whose musical work "The Well-Tempered Clavier" solved the old problem of instrument tuning with respect to the physical constraints of harmonic scale tonality. Further, in the following section, we will introduce the Artificial Painter, which can be considered a good example of how today's technical revolutions might affect art.

14.4 ELECTRONIC ART

So far we have highlighted two kinds of art action: one related to ideas, language, and meaning (i.e., "immaterial" art) and one related to physical action by the artist and to art technologies (i.e., "material" art). We then depicted two specific social spaces where electronics plays a role, the machine (hardware) and the inner workings of the machine (software). So, given all this, what are the implications of electronics and how are artists going to use it?

Many of us have the feeling that we are at the beginning of a new technocultural revolution, which will bring us to new frontiers of knowledge and will change our lives significantly. Electronics is leading the revolution, which, it seems, will involve every possible category of human artifact, right up to human thoughts—in short, the whole of society. Art will not be external to this process; rather the contrary, it might even have the hard task of somehow, at least partially, guiding it. As the first section argued, when discussing Pythagoras's discoveries, artistic movements, philosophically speaking, "fight" the abnormalities of science and technology by modeling, shaping, and finally bending them to real human needs. Art acts together with religion (when the two can be separated) as a guardian of the conscience of the human race—as demonstrated by George Orwell.

So let us return to art and try to understand the nature of the new frontiers introduced by technology, how art might influence them, and how art itself might be influenced by them. Let us try to describe the essential technoartistic scenario.

14.4.1 A New Electronic Space

Besides the social space occupied by electronic software and hardware, electronics also occupies a physical space of its own that we will introduce in this paragraph. As suggested above we can distinguish between "material" and "immaterial" art. Where do we find this distinction in electronic art? Although the categories today seem outmoded, the history of ideas and philosophy has identified five main artistic disciplines: painting (we include here all the visual arts such as photography and cinema), dancing, sculpture, music, and poetry. As a consequence, you can call yourself an artist if you deal with and excel in at least one of these disciplines. What does this mean? A closer examination shows that in order to be considered an artist you should be able to move in a smart, aesthetic, and emotional way in one or more of five different spaces. The painter has always been the one who smoothly steps out across a canvas surface. The sculptor sharply slides into rock, wood, and marble. The musician flies in among sound waves. The dancer moves tenderly in body space. Finally, the writer jumps surprisingly between words. Each has a well-defined and recognizable role; for each there exists a well-defined physical place where skills can be demonstrated and compared, allowing the identification of true artists—expert "walkers" in their own specific space. These are what we called "material" artists. But what has this to do with electronics?

This is a difficult question. It seems to us that electronics and what we call software and hardware have defined a new kind of space where the "old artists" would find it hard to fit. The boundaries of this new space are, of course, rather loosely drawn. But the deeper you step into electronic space, the closer you get to identifying what we mean by an electronic artist.

14.4.2 The "Material" Electronic Artist

The space in which electronic art takes place is new space, and even though we usually identify it with a PC screen, it is, in reality, much more than that. It is not a printout or a "wave file." We can play the violin or a synthesizer and still remain traditional musicians; the space we are moving in is the traditional one for musicians. In the same way, we will soon be able to design an amazing three-dimensional file format and print it out on a 3D printer or design a new robot. In both cases we can still consider ourselves sculptors. In the same way, we can work out a poem on a word editor (perhaps a hypertext editor), without this having anything to do with real electronic art.

What we are trying to say is that electronic art works in a different space. Electronic art (as opposed to the traditional arts, mediated by electronics) can be identified by the physical space it occupies, by the kind of matter it shapes. This is closer to the underlying electronic logic (if not to the CPU and to specific hardware circuitry). The space defines the way we can move in it, the ways in which we can exert or lose our control. Yes, of course, we act in the space, and like any other action in our universe, what we do may be in some way audible, touchable, visible, and so on. Still, this might not be the focus of the artistic action or the most important thing for the artist—the walker in this new kind of space.

Let us take an example. We sit in front of a PC and start writing code, say, Java code. We are going to write a genetic algorithm (Holland 1975; Mitchell 1997; Goldberg 1989). Our algorithm takes shape, and we write nice functions that give us back amazing results and, maybe, some errors that we don't care about since we, in this context, do not look upon ourselves as scientists (scientists cannot neglect uncertainty in the same way artists might and usually do). OK, now that we have billions of nice numbers coming out of our debug console, what shall we do with them? Say that we plot them on a graphical display. What will that make us? Will we be painters? And what if we plot them onto a sound file? Does that make us musicians? What if we do both? What if we do both simultaneously? Well, we think that it does not matter at all. Insofar as we have a broader sense of art, we seek the best possible representation for our numbers—the one that best shakes people's hearts. If, on the contrary, we lack this sense of "material art," we will end up like Pythagoras. In both cases we might presume to call ourselves artists (or scientists). What we would not do is call ourselves musicians or painters.

If you agree with this, you will also agree that we still need a definition of what we mean by an "electronic artist." Before reaching such a definition, let us make one further consideration.

14.4.3 The "Immaterial" Artist and the Uses of Electronics

The sort of process we have just described is not of course the only possible use for electronics. We can bend electronics to our aesthetic tastes and our will without physical contact—without programming or assembling a circuit. We could, for example, hang a computer from the ceiling. This might still be considered art. It is a kind of art that has quite a lot to do with electronics, but is not electronic art and has nothing to do with it. This is a kind of art that does not focus on the object but relates only to the meanings that objects and facts carry with them. It has nothing to do with science, not directly. This is what we called here "immaterial" art.

14.4.4 Example—The Artificial Painter

As an example, we will introduce the Artificial Painter, software that we developed to explore and develop electronic art. In the context described above, the Artificial Painter should be regarded as a tool for the eletronic, "immaterial" artist. The Artificial Painter is based on artificial neural networks and evolutionary algorithms. At the beginning of each evolutionary session, the program generates 16 random images that are used as "seed images" for the real computer painter to start to work. One can choose four of them, which the computer treats first as individuals and immediately after as parents that produce 16 new individuals constituting a new generation. The painter can zoom out each of the images, carefully look at them, and observe tendencies, trends, changes, and interactions of composition, color, texture, form, perspective, and their respective interactions. The only parameters one can control are which individuals will become parents and generate a new generation of 16 individuals under a mutation percentage we define for a particular generation. We can assign a different mutation percentage for each generation.

The idea to develop a computer tool for artistic expression originated from the field of artificial life. Artificial life is concerned with the study of life-as-it-could-be—for example, on computers but also in robots—rather than life-as-we-know-it from nature. Artificial life research can be performed, for instance, by studying the evolution and behavior of simulated animals, called *animats,* in computer simulations. It is these kinds of artificial life computer simulations that form the fundament for the Artificial Painter.

A single animat in a population of such animats is represented by an artificial neural network. An artificial neural network models a natural nervous system and consists of neurons and synapses, just as the natural nervous system does. In the case of an animat, the artificial neural network gets input from the environment, propagates this activation through the network, and produces an output that can be interpreted as a motor action. The output produced given a specific input from the environment will depend on the architecture of the artificial neural network—in other words, it depends on the animat's "brain." The advantage of using artificial neural networks instead of more traditional computer programming techniques is that the artificial neural networks can be trained (e.g., by trial-and-error or reinforcement) to learn to perform specific tasks, just as we humans can learn new skills or we can teach our pet to perform specific tasks.

An example of an artificial life experiment would be to place animats in a simulated environment, in which a number of food elements are distributed at random positions. That is, the environment is a grid world in which an animat occupies one cell and food elements occupy other cells. A goal of the artificial life experiment could then be to evolve the animat to perform the task of navigating around in the environment and collecting as many food elements as possible. This would demand the evolution of a suitable brain (i.e., artificial neural network) that responds with an appropriate motor action (i.e., displacement of itself in the grid world) to each possible sensor activation, where sensor activation could be perception of angle and distance to the nearest food element.

In order to develop such a food collecting behavior of the animat, we used a so-called evolutionary algorithm. An evolutionary algorithm works on a whole population of animats and models the Darwinian evolution theory of "survival of the fittest." Initially, the evolutionary algorithm constructs a population of random animats (that is, it constructs random artificial neural networks) that are put in the environment sequentially one after another for a given time period. When an animat is put in the environment (i.e., the grid world), it will respond to its sensory input based on the characteristics of its artificial neural network. Since the network is constructed at random, the animat might respond very badly on the task of navigating around and collecting food elements. Based on the animat's performance (how many food elements it collects) during the specific test period, it will achieve a score. This score is called the animat's fitness. Then, the second individual animat of the population is put in a similar environment and is tested for its fitness. After this, the third animat is tested, and so forth. Hence, each individual animat of the population will achieve a fitness score. Now, the subset of the population that obtained the highest fitness scores is allowed to reproduce—hence the term "survival of the fittest." Each of the individuals in the selected subset is copied a number of times to make that number of children. However, the children are not perfect copies, since biological operators such as mutation are applied during reproduction. During mutation, a small part of the copied artificial neural network undergoes a change. Therefore, the children inherit most of the characteristics from their parents, but they also differ a bit from the parents. We can call the first set of randomly generated animats generation 1 and the reproduced children generation 2. When generation 2 has been constructed via selective reproduction, it can be tested and each individual animat obtains a fitness score. The best subset of generation 2 is selected to reproduce in order to generate generation 3 of animats. In this way, the evolutionary algorithm generates generation after generation, which each are tested and selectively reproduced.

Scientific studies have shown that such evolutionary algorithms can successfully evolve artificial neural networks to solve a range of different tasks, including finding appropriate "brains" for the animats to solve their food-collecting task.

However, a major drawback with this approach is that, for each specific task, a fitness function that determines the fitness score of the individuals has to be designed and described in mathematical terms. In a sense, the mathematical fitness function decides what a good "solution" is. Especially, it is a drawback when we want to use this approach in developing artistic expression, since the aesthetic value of an art piece cannot be described as a mathematical fitness function and further might depend on individual taste.

Therefore, we have developed a so-called *user-guided evolutionary algorithm* (used both in art, robotics, and educational software). Instead of having a global mathematical fitness function that selects which individuals to reproduce generation after generation, we allow the user to decide, at each generation, which of the proposed individuals should reproduce from that generation. This is done by representing the whole population on the computer screen, and then allowing the user to select a subset of the population by clicking on individuals with the computer mouse. Hence, there is no need for a mathematical description of fitness, and it is the user who decides, at each instant, which individuals should be allowed to reproduce. There are no constraints on the selection. Different users might choose differently, and a single user can change preferences from generation to generation.

In the case of the Artificial Painter, we represent a population of 16 pictures on the computer screen, and users are allowed to select 4 of these to reproduce according to their own aesthetic evaluation. Each single picture is constructed by an artificial neural network that, like the animat, is placed in a grid world with two landmarks placed at different positions. In fact, the idea of the Artificial Painter emerged from such artificial life simulations, in which we wanted to evolve animats to perform landmark navigation. In order to study the characteristics of the single artificial neural network, we found inspiration from the diagnostic and neurobiological techniques for brain scanning, such as positron emission tomography (PET), computed tomography (CT), and single neuron records. With these techniques, doctors can scan the brain and get images of neural activation of different parts of the brain. We used a kind of single neuron records technique on the animats to obtain images that could allow us to analyze what was going on in the animat's "brains."

We place an individual animat in the first cell of the grid world and record the activation of a specific neuron in the artificial neural network. The activation of the neuron is dependent on the sensory input (angle and distance to a landmark) and the network architecture. This calculated activation is mapped into a color. We have a total of 256 colors, so the range of possible activations of the neuron ([0.0,1.0]) is divided into 256 intervals that each corresponds to a specific color. Then, we place the animat in the second cell of the grid world, record the neuron's activation, and map it into a color. By doing so for all the cells in the grid world, we obtain a color for each cell in the grid world. The image of the colored grid world (i.e., the neuron activation at each position of the grid world) is shown on the computer screen. This constitutes one Artificial Painter picture, and the whole population of 16 individuals runs through the same procedure in order to produce 16 pictures on the computer screen. However, in order to obtain a larger variety in styles of pictures, we allow not only the artificial neural network architecture to evolve, but also the colormapping and landmark positions in the grid world are allowed to evolve. In a sense, we can say that artificial neural network architecture, colormapping, and landmark positions are included in the individual animat's genotype—it is the genotype of an animat that evolves. Hence, when a picture is selected to reproduce, mutation will also work on the colormapping and landmark positions so that color and form in a picture may change from one generation to another.

We also include a neuron output activation function in the individual animat's genotype. This function determines the output activation of the neuron that is then mapped into a color, as explained above. The net input to the neuron is processed by a mathematical function that can be formed by a combination of functions such as sum, logistic, exponential, cosine, sine, and so on.

In total, with artificial neural network architecture, composite activation function, color palette, and landmark positions in the individual animat's (i.e., picture's) genotype, we obtain an incredible range of possible outcomes. It is then the painter who, by indicating his or her preferences, guides the development process toward a specific type of picture (see Color Plate 8).

With the Artificial Painter, analysis of inner trends and tendencies within a certain image, prediction of outcomes under particular mutation percentages, and reflection on possible interactions between texture and form, form and perspective, light and texture, color and composition, and so on became elements that are deeply rooted in each other. The painter does not have in his or her hand brushes, colors, and other painter's tools. Imagining developing worlds through the interactions of their features and appearances creates internal capacities for algorithmic perception, reflection, and interaction of basic visual features. Amazingly enough, a lack of firm tools helped a lot in developing a sort of etheral negative expertise in intuition, algorithmic perception, dynamic and interactive aesthetic reflection on interactions, and transcendings of basic visual and spatial features.

14.5 ALIVE ART

Electronics and art have already developed a significant dialogue. As a result, the number of artistic movements that make use of electronics grows day by day, rather like biological species during the Cambrian period. One of the most evident effects of this "Cambrian explosion" is confusion.

This is fair enough. The growth of electronics affects other arts and produces art itself. This in turns modifies the use of electronics in art and is influenced by it, in a sort of endless loop. Nevertheless, before introducing the concept and definition of "Alive Art," let us first refer to a number of electronic art movements and make a few general considerations.

14.5.1 Other Artistic Movements Based on Electronics

The most famous and widespread electronic art movements are "electronic art," "digital art," and "computer art." These "old" definitions, despite their popularity and historical importance, have, in our opinion, lost much of their theoretical significance. This kind of approach to electronic art is too vague for the current situation. What they express is the concept that art has something to do with electronic, digital, or computer-based materials. It is self-evident that we need clearer analytical distinctions.

The first thing to be noticed is that, most of the time, there is a quite brutal distinction between artists who use software and those who work with hardware. Considering our analysis of electronics' social context this is not surprising. On the software side we have artists who use various kinds of artificial life (Langton 1992) algorithms to paint a PC screen (or its equivalent) or (more rarely) to generate sound. Hardware artists, on the other hand, create (primarily) cybernetics-based works of art with a few basing their efforts on pseudo (i.e., nonautonomous) robotics. Among these artists the most productive have been (for solid economic reasons) those engaged in the design of algorithms for sounds and images. There are dozens of proposals of this kind: artificial life art, genetic art, generative art, evolving art, evolutionary art, organic art, fractal art, and so on (Sims 1991; Lund, Pagliarini, and Miglino 1995a), of which the above-mentioned Artificial Painter is an instance.

The common feature of all these movements is that they are based on algorithms that in some way produce their own rules and that generate their own behavior. In our opinion, the most appropriate descriptions of this work might be "genetic art" (because of the connection to Darwin), "artificial life art" (because of the connection to Langton and like thinkers), and "generative art" because of the link to Chomsky. It should be observed that, as far as the authors know, none of these artistic movements have concretely applied their basic concepts to hardware, and modeled hardware and software as a unified whole. In other words, most of the time it is the software that operates the artistic transformation while the hardware is "hand-crafted" with at most the ability to reiterate some limited "movements." This is crucial. Although technically possible, there has, in practice, been no revolution in the material structure underlying this kind of art.

In hardware-based electronic art things are quite different. There is not room here to go into a detailed analysis. It is nonetheless necessary to make a distinction between early cybernetics and later cybernetics or movements like that of the kinetic sculptors. Earlier cybernetics artists were, from all points of view, the precursors of computer science and lifelike algorithms. Indeed, cybernetics, originating prior to the transistor, was successful, right from the beginning, in creating electromechanical analogies to living systems. These artists made direct use of electromagnetic fields in art. Today cybernetics artists, kinetic sculptors for example, are quite different. Not only do they give a bigger role in computer-based technologies and a lesser one in electromagnetic fields, they also make heavy use of technicians, computer scientists, and engineers in their artistic production. In other words, they are moving far away from the original type, and philosophy, of the artist and opening themselves to a conception of artwork as work by a team. Philosophically speaking, this drastically changes the artist's attitude; the use of human material becomes, of necessity, part of the act itself—a fine challenge. Politically speaking this might be a constraint. Materials, including other humans, are very expensive. For this reason, many of these works of art are financed by companies. Inevitably, the artist loses his or her freedom. A good example of a similar tendency is the cinema, where the artistic component is weakened by the need to generate profit. This is a danger that we should keep clear in our mind and that leads us to additional considerations. As we said earlier, the situation is a little confused, and the overall scenario is so dynamic that more time is needed before the relationship between art and electronics becomes fully clear. We can nonetheless attempt to outline this relationship.

14.5.2 Alive Art

Art history tells us we are following a path that leads toward immateriality. As a consequence, visual artists, for example, have moved from painting to photography to cinema to computer graphics, and from figurative painting to impressionism to futurism to generative art. It appears as if there is a need for an artistic discipline that underlines the restless aspect of representation (in this case visual representation). We are searching for a meaning of dynamics that is not only moving and changeable but that can go further. For these reasons it seems that we now have the possibility of shaping a new art movement or approach, whose medium is mostly, but not necessarily, electronics. This is *Alive Art*. Alive Art should be a discipline where the dynamical aspect of the work of art is crucial, if not essential.

Further, the use of the term "alive" stresses that works of "Alive Art" should be ever changing as well as ever moving. Things that are alive can die—and they can also react. As a consequence, the characteristic of Alive Artworks would be perpetual change (which can also lead to extinction or death) and interactivity. There is in this definition a strong drive to define an artwork that is, conceptually speaking, immaterial, abstract, difficult to put in words. This should be an art movement that has a strong relation with the deepest aspect of life. This is not, or not only, change but perpetual change, which is not only action but constant action and reaction, which leads not only to changes in life but also, at times, to the end of life (i.e., to vulnerability). If we were asked: "What kind of changes are you talking about?" the obvious answer would be "In Alive Art, as in life, there should be unpredictable, as well as predictable, changes." That is, in our opinion, the way to go. It is the course of art.

14.5.3 The Aliver

As stated above, Alive Art would be recognizable by perpetual change, interactivity, and by the vulnerability of the works of art it produces. But how should we identify the performer of Alive Art, the Aliver? In Section 14.1 we introduced the social context where electronics moves, while in Section 14.3 we suggested that electronics has introduced a "new" space where the artist can "walk." We went on to emphasize the risks the artist is taking (i.e., loss of artistic freedom). These two elements are relevant to understanding the Aliver. The Aliver should move easily in the social contexts and spaces of relevance to electronics. Although we agree that, in some recent artworks, human material is more or less necessary, we also believe that the new electronic artist, in this case the Aliver, should, like a sculptor, be as close as possible to the materials he or she is working with—relying as little as possible on other human beings. This is crucial, not only for the "political" reasons we discussed above, but also because of the relationship the artist builds with the "life" of the object. In brief, the Aliver should be as close to the software as he or she is to the hardware (or whatever material he or she uses). If the Aliver concept is taken to its extreme, it is the Aliver who kick-starts the life of the work of art (i.e., an object that changes constantly), allowing it to reach a point where it (e.g., a robot) can produce art on its own.

At least from an electronic point of view, this is not just a vision. It is, at least partially, already happening in robotics (as shown by the robot musicians and robot painters in Color Plates 9 and 10). It was done by the inventor of the Internet. So, the nearer you get to the electronics, the closer you are to being a pure Aliver. The techniques that one of today's Alivers might use—range from artificial life techniques, such as genetic algorithms (Holland 1975), neural networks (Rumelhart and McClelland 1986; Parisi, Cecconi, and Nolfi 1990), and cellular automata, to techniques from electronic engineering, such as sensors, motors, chips, and lasers, to even biological techniques, such as genetic engineering, microsurgery, and neurosurgery. The Aliver might design both the outer and inner body of the work of art itself and of those who interact with it. The Aliver should be no more outside the technological process than the artist is outside the artistic process. Indeed, to use and deeply know new techniques and technologies is one way, maybe the best way, to give the right emphasis to what society is, to what it is becoming, and to the meanings it is carrying along into the third millennium.

14.5.4 The "Alive Art Effect"

In line with the above definition and with the categories previously discussed (i.e., the internal and external aspects of the machine), we can try to imagine two kinds of Alive Artwork. An example of what we called "outward-looking" electronics, which acts in the real world and obeys real-world physics, might be a robot artist with the ability to evolve and modify, time after time, its own body or, at least, its electronic circuitry. This might consist, for example, of a neural network controller, which adapts to past interactive experiences with objects, animals, and humans. An example of "inward-turned" electronics could be an Alive Artwork suggested by some expressive phenomena that spontaneously emerged from the Internet. Or imagine an interactive genetic programming algorithm with the ability to modify itself every time a user uses (or downloads) the software. Both these artworks would be lifelike objects whose creator, the artist, having designed and set in motion their generative principles, would lose control over his or her own work. That would be what we call the "Alive Art effect."

14.5.5 Example—LEGO Robot Artists

Some LEGO MINDSTORMS robots are good examples of another way of using hardware in electronic art. For the LEGO MINDSTORMS European RoboTour 1999/2000, LEGO developed a number of robots to perform different shows, such as a bungee jumper, a fashion show, a wall climber, and so on. Here, we will focus on two of these setups: the robot musicians and the robot painters.

The LEGO robot painters make an abstract painting on a canvas of approximately 1.0 m × 1.7 m. There are four such painting robots, and they are all based on LEGO MINDSTORMS, using the RCX (Robot Control System) to control the actions of the painters. For instance, one robot painter spray-paints, one paints with a broad brush, and one splashes a big blob of paint on the canvas once in a while. The robots are autonomous, but a professional artist can overwrite the autonomous actions with steering commands from a joystick. But the artist can only control one robot at a time (see Color Plate 9).

The LEGO MINDSTORMS robot musicians are an orchestra that plays one of Strauss's most famous waltzes, "The Blue Danube." One robot plays the drums, a couple of robots play the xylophone, one robot plays a stringed instrument, and so forth (see Color Plate 10). There is even a robot conductor. One of the most striking characteristics of the robot musicians' performances is that their "execution," due to technical reasons, is *never* the same. It is, in other words, very much like the performance of human musicians, and that leaves the audience breathless, most of the time. Apart from that, the robot musicians are a good example of what we mean by "aliveness" or at least ever-changeability.

A further evolution in this type of robot performer—performers like the robot musicians and the robot painters—might be the increment of the degrees of freedom that the robots themselves could have in "interpreting" musical and graphical patterns. Imagine a robot painter that instead of being controlled by a joystick could be "influenced" by an artist's input or feeling (e.g., related to the artist's physiological activation of the neural activity of real nerve cells). Of course, to advance the concept further, we could imagine a completely autonomous robot painter. Another good future application example could be robot musicians that according to an intelligent algorithm, are able to interpret both the written and the unwritten music and the feedback from audience reactions (e.g., by reading input from a camera).

The whole process would lead to an evident "Alive Art effect." In other words, the improvement in electronic tools' and devices' "actions" seems to be directly proportional to the improvement of artificial intelligence algorithms.

The whole process might lead to a relatively consistent autonomy of robot action and, as an obvious consequence, the use of those robots for mediating art expressions.

14.6 CONCLUSIONS

In a human social context electronics has an inner and an outer aspect—the software and the hardware, respectively. Moving from this background we have investigated the relationship between art and technology. Technology or science produces deep changes in art. We have cited, as an example of this, the influence of photography on painting. We then highlighted the current relationship between art and electronics with respect to what we called "immaterial" and "material" art. We emphasized that while the exponential growth of electronics requires science-oriented artists, the explosion of technical knowledge also calls for teamwork. We pointed out the dangers to which this can lead. We summarized the current state of electronic art and outlined the characteristics of a new art form—"Alive Art"—characterized by perpetual change, interactivity, and vulnerability. We presented examples showing what an Alive Artwork might look like and the way in which the performer of Alive Art might lose his or her artistic authority. While Benjamin, in his time, saw a loss of spatiotemporal unity in art (i.e., pointing to the problem of the reproducibility of the work of art), electronics seems to undermine the very identity of the artist.

REFERENCES

Benjamin, W. (1995) Das Kunstwerk im Zeitalter seiner techniscen Reproduzierbarkeit. In *Schriften.* Suhrkamp Verlag.

Goldberg D. E. (1989) *Genetic Algorithm in Search, Optimization, and Machine Learning.* Addison-Wesley.

Holland, J. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press. (Or MIT Press, 1992.)

Langton C. G. (1992). Artificial Life. In L. Nadel and D. Stein (eds.), *Lectures in Complex Systems,* SFI Studies in the Sciences of Complexity, Lect. Vol. IV.

Lombardo, S. (1983). Percezione di Figure Grottesche in Alcune Strutture Casuali. In *Rivista di Psicologia dell'Arte,* Anno V, nn.8/9, giugno e dicembre.

Lund, H. H., J. A. Arendt, J. Fredslund, and L. Pagliarini (1999). Ola: What Goes Up, Must Fall Down. In *Proceedings of Artificial Life and Robotics* (AROB'99), ISAROB, Oita.

Lund, H. H., O. Miglino, L. Pagliarini, A. Billard, and A. Ijspeert (1998). Evolutionary Robotics—A Children's Game. In *Proceedings of IEEE 5th International Conference on Evolutionary Computation,* IEEE Press.

Lund, H. H., and L. Pagliarini (1999). Robot Soccer with LEGO Mindstorms. In M. Asada and H. Kitano (eds.), *RoboCup'98,* LNAI 1604, Springer-Verlag.

Lund, H. H., and L. Pagliarini, (2000). RoboCup Jr. with LEGO Mindstorms. To appear in *Proceedings of Int. Conf. Robotics and Automation 2000* (ICRA2000), IEEE Press.

Lund, H. H., L. Pagliarini, and O. Miglino (1995a). Artistic Design with Genetic Algorithms and Neural Networks. In J. T. Alander (ed.), *Proceedings of 1NWGA,* Vaasa University.

Lund, H. H., L. Pagliarini, and O. Miglino (1995b). The Artificial Painter. In *Abstract Book of Proceedings of Third European Conference on Artificial Life,* Granada.

Miglino, O., H. H. Lund, and M. Cardaci (1999). Robotics as an Educational Tool. *Journal of Interactive Learning Research* 10(1): 25–47.

Mitchell, M. (1997). *An Introduction to Genetic Algorithms.* MIT Press.

Pagliarini, L., A. Dolan, F. Menczer, and H. H. Lund (1998). ALife Meets Web: Lessons Learned. In J. C. Heudin (ed.), *Proceedings of Virtual Worlds,* First International Conference, Springer-Verlag Press.

Pagliarini, L., H. H. Lund, O. Miglino, and D. Parisi (1996). Artificial Life: A New Way to Build Educational and Therapeutic Games. In *Proceedings of Artificial Life V.* MIT Press/ Bradford Books.

Parisi, D., F. Cecconi, and S. Nolfi (1990). Econets: Neural Networks That Learn in an Environment. *Network* 1(2):149–168.

Rumelhart, D. E., and J. L. McClelland (1986). *Parallel Distributed Processing. Explorations in the Microstructure of Cognition.* MIT Press.

Sims, K. (1991). Artificial Evolution for Computer Graphics. *Computer Graphics* 25(4):319–328.

This Page Intentionally Left Blank

15 CHAPTER Stepping Stones in the Mist

Paul Brown

15.1 INTRODUCTION

This chapter provides an idiosyncratic and nonrigorous account of my work as an artist who has been involved in the field now known as artificial life for over 30 years. To give the reader some context I will begin with a few opinions that define my position within the visual arts (which is far from the current mainstream) and then go on to describe early influences from the 1960s and 1970s that have framed my involvement in the field of computational arts. This includes some examples of my work from this period. The latter part of the chapter describes my working methodology and includes examples of my more recent work. The chapter ends with some speculations about where I may go in the future.

The title is a metaphor for my self-view as an artist, and individual. A long time ago I stepped off the bank of a misty river or lake and onto a line of stepping stones. Now, many years later, the stepping stones are shrouded in the mist. Those behind me are dimmed by the mists of memory, and those in front are hidden by the mists of uncertainty. The one in front of me is quite clear (as is the one behind), but then they quickly fade as they progress. I have no idea what lies on the further bank, or indeed if such a shore even exists! Memories of the bank I left are now long eroded.

I only really know where I am at this moment or, perhaps, where I have just been.

15.2 ON MY APPROACH AS AN ARTIST—A DISCLAIMER

Thanks to my longstanding interest in computational systems as a medium for the visual arts I have been relegated to the fringes of the arts mainstream for most of my career. The role of outsider is one that I enjoy, and I was somewhat disturbed when the global art mafia appeared to be acknowledging the computer-based arts in the early 1990s. In retrospect I had no need to fear. The mainstream's adoption of this area of work was, and is, extremely parochial, one-dimensional, and, dare I say, paranoid.

Work that uses computer-aided tools (productivity enhancers based on traditional tools and methods) has been adopted and, for a brief time at least, became exceptionally fashionable. However, the concept of the computational metamedium, to quote Alan Kay's term (Kay 1984), as a unique new paradigm for the arts quickly fell prey to the "no skills please—we're postmodernists" kind of rhetoric that the international contemporary arts scene uses to defend their position whenever it is threatened.

The paradigm shift that is foreshadowed by the computational arts is, quite correctly, perceived by the holders of the status quo as a significant threat to their jurisdiction. They are building barricades of rhetoric to shore up the crumbling foundations of their glass menageries. As I have commented elsewhere, a revolution is in process in the arts, although, given the extreme conservatism of the discipline, its cliquish nature, and its lack of any kind of quantitative foundation, I have little expectation of a resolution in the near future. When I was 20 I expected that the revolution would be over long before the time I was 30. Now I'm 52 I will be surprised if it's resolved in my lifetime. I nevertheless remain optimistic that it will be resolved, and in my favor!

I hold dear to many unfashionable concepts in this brave new postmodern world. A favorite is my belief that the culture vultures of the arts mainstream have completely confused postmodernism with their own brand of ultraconservative late-modernist rhetoric.

Here, however, are some more productive opinions:

- The artistic mind is a "butterfly" mind that can fly from flower to flower, from source to source, with little respect for logic or scholarship. The result is a grand synthesis formed at a meta- or preconscious level. From this, it follows that:
- The visual arts are beyond language, beyond conscious processing, at least when they are created.[1] From this we can expect that:
- Theory does not (necessarily) inform creation, although creation, of necessity, informs theory. In believing that living art (art that is still in the process

---

[1] The studio/production basis for the visual arts often means that they sit uneasily within scholastic institutions like universities where their studio component is often undermined in favor of theory. This is partly due to economic rationalism (studio is one-to-one and expensive, theory is one-to-many and cheap) and partly due to the academic pressures of the university tenure and promotion policies (which favor theory).

of being created) is by definition beyond the linguistic mechanisms that critical theory demands, I nevertheless acknowledge that dead art (like Dada, which is a complete corpus and cannot now be modified) is susceptible to theoretical deconstruction and analysis. Perhaps Dada can be totally described by language, though I suspect that Godel's concept of incompleteness will apply here as it does in any defined and rigorous domain. If we want to learn about Dada, it will probably help us if we look at some Dadaist artworks, although I'm not convinced this is absolutely necessary. If we want to learn about a living art process, then we are obliged to look at and/or interact with the work. It is the only portal for understanding that we possess.

I also suspect that there may be analytical tools, like Charles Saunders Peirce's semiology or George Spencer-Brown's Laws of Form, that may have more success in their application to the living arts. My suspicions are based on the intuition that these tools may share, or overlap with, the metalinguistic domain of visual artistic creativity.

- The medium informs the work, and skill with the medium determines the quality of the work. This is a very unpopular point of view at present and considered a legacy of high modernism's "truth to the medium." I challenge all critics to write a poem in a language with which they are not familiar that a native speaker of that language would consider acceptable. It doesn't have to be good, just acceptable. Random "cut-ups" and computer translations don't count.

15.3 MAJOR INFLUENCES

Anton Ehrenzweig's *Hidden Order of Art* was recommended to me soon after it was first published in 1967. As a young art student it meant little to me, and it wasn't until I had become interested in system or procedural art in the early 1970s that it made much sense. Ehrenzweig was a psychoanalyst who has been credited, by Anthony Storr, with being responsible for the major revision of the discipline that redirected it from the mainly pathological focus of Freud to a more creative and celebratory emphasis. I suspect that Ehrenzweig may himself (had he still been alive) have credited Marion Milner with this transition.

His first book, *The Psycho-Analysis of Artistic Vision and Hearing: An Introduction to a Theory of Unconscious Perception,* is a flamboyantly unstructured and almost unreadable cry of "eureka." *Hidden Order,* if he had lived to complete it, would have been his masterwork. It was published soon after his death in an unfinished format, unindexed, and after a flurry of interest disappeared from view before reaching a second edition. After a long hiatus it reappeared and is still in print from University of California Press.

Ehrenzweig overcame the significant problems that had undermined Freud's analysis of the overt content of the artwork by analyzing instead its structure and, in particular, the structure of the creative process itself. This was of course a period when Abstract Expressionism—random or "subconscious" mark making—was the dominant model for the visual arts. Ehrenzweig proposes three major stages in the creative process: an initial rejection of unwanted or repressed material, followed by an "oceanic" engagement and synthesis of the material, and a final reintegration or "reification" of the material at a conscious level.

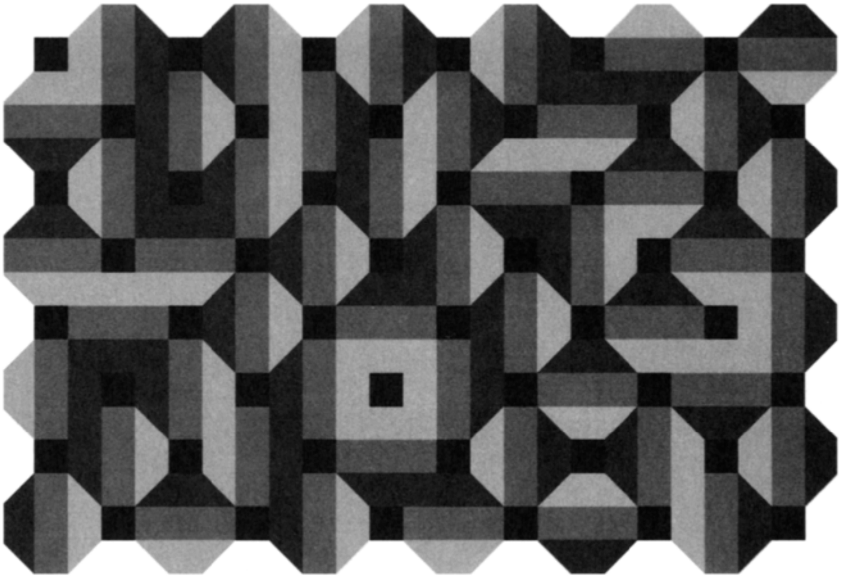
I read *Hidden Order* at one sitting and remember my excitement and agitation when I had finished it. Within a couple of hours I had devised a procedure that I thought might be capable of "testing" Ehrenzweig's hypothesis. It involved replacing Ehrenzweig's initial rejection stage with a system for positioning tiles according to the output of a random number generator—a die! This appealed to me because I was suspicious of all references to a subconscious and then, as now, was extremely sceptical of the concept of art as self-expression (at least in the emotional sense of the phrase).

Figure 15.1 shows recent reconstruction of the first image I made to test Ehrenzweig's hypothesis. It used octagonal tiles, and the four orientations were dictated by flicking the pages of the book and using the last digit of the page number modulo 3—I didn't have a die at that time! The black squares indicate holes.

My conscious motives (as I remember them) were involved with issues that obsessed me at the time like removing myself from the work and objectifying the art-making process. Issues that had, in that period of history lead to Minimalism, Conceptual Art, and Art Language. In retrospect my achievement was the establishment of a personal methodology for creative production that has governed my work ever since.

Around that same time George Spencer-Brown's *Laws of Form* was recommended to me (Spencer-Brown 1969). I found its clinical notation intimidating, and in consequence, I read a teach-yourself format book on symbolic logic. On returning to the *Laws* I discovered my concern had been unnecessary. Spencer-Brown leads his readers step-by-step through his calculus and toward his conclusions. More recently it has been described to me as a "boundary grammar" since it deals with the ideas of distinction and of crossing.

Although I remain convinced that *Laws of Form* is one of the most important books I have ever read, I am not aware of any direct influence it has had on my work. I remain surprised about this and often carry a copy around with me so I'm ready for the breakthrough when it occurs! Spencer-Brown's clinical and



15.1 FIGURE Testing Ehrenzweig's hypothesis.

precise methodology has certainly influenced me. Ironically symbolic logic, which I perceived at the time as merely a spin-off or perhaps more correctly a portal to the *Laws,* has had a profound and ongoing influence. This is probably because of my immersion in computational methods and their foundation in logic and formal languages.

Another book that was a major influence at the time was Charles Biederman's classic *Art as the Evolution of Visual Knowledge* (Biederman 1948).

Since 1968 I have been fascinated by the structure of a classical Chinese text called the *I Ching or Book of Changes* (Wilhelm 1923/Baynes 1967). It is a two-state system of broken (yin) and unbroken (yang) lines that influenced Leibniz, who developed the European version of binary notation. The basic "word" is a three-bit trigram. The eight trigrams are multiplexed together to form six-bit hexagrams, which index the 64 chapters of the book. Via changing lines (bits that can flip from yin to yang) any chapter can change to any other and so the combinatorial permutations of the book total 4,096. The book is believed to have first appeared around 1800 BCE. It is possible to interpret the book as a symbolic cosmology that derives the three-dimensional Cartesian universe by repeated subdivisions (the trigrams) and then populates it with agents (the hexagrams). This process is echoed in the opening stanzas of the *Tao Te Ching:*

One gives birth to two [the yin and yang]
Two gives birth to three [the trigrams]
Three gives birth to the myriad creatures [the hexagrams]

In the 1970s I also became aware of the work of the System Art Group. Several members taught at the Slade School of Art, where I was to study from 1977 to 1979. None of them, at that time, used computers, and although I cannot think of any direct influence their work has had on me, it was certainly reassuring to meet others with a similar mindset!

In 1970 I read Martin Gardner's column "Mathematical Recreations" in *Scientific American* (Gardner 1970) where he described John Horton Conway's "Game of Life." For several months I persevered with large sheets of graph paper layed out over the floor of my home. Pencil, paper, and eraser were too limited, and I had to wait four years until I found my "ideal" tool—the digital computer. However, this initiated my fascination with cellular automata (CAs) that continues to this day.

The final influence I will mention from that brief but formative period between 1968 and 1972 is the exhibition Cybernetic Serendipity, which was curated by Jasia Reichardt and held in 1968 at the Institute of Contemporary Art at its then-new premises on the Mall (Reichardt 1969). It was the first historical review of artists using computers. I was fascinated and returned to London in order to spend a second day at the show. Although I was attracted to the idea of working with computers, my attempts to get involved didn't come to anything. Then the Polytechnics were formed, and it was possible for me to enroll, in 1974, as a mature fine arts student at Liverpool Polytechnic (now John Moores University) and then spend most of my time in the mathematics department learning Fortran (on an ICL 1903a) and in engineering discovering PAL3 Assembler on their DEC PDP 8. Despite this, in 1977, I was awarded a first-class honors degree in fine art.

15.4 HISTORICAL WORK—1960S AND 1970S

As a young art student at Manchester College of Art in 1965 I had the choice of being pigeonholed as either a painter, sculptor, or printmaker. I survived three years of boredom before dropping out to cofound the lightshow Nova Express, which toured the North and Midlands of England for several years. In addition to the leading bands of the day like Pink Floyd, The Nice, and Canned Heat, we also played with contemporary music, dance, and performance groups like Music Electronica Viva and Meredith Monk and The House.

About that time John "Hoppy" Hopkins introduced me to video, and, in collaboration with the musician and composer Michael Trim (who had one of the first AKS digital audio synthesizers) and an engineer (whose name I am sorry to admit I have forgotten), we made several primitive video synthesizers. These worked on principles of feedback and utilized both electronic and mechanical components (like rippled glass filters). Although most of our work was intended to be played live, we did make a few tapes, and one "Mandala" was included in the UK's first major retrospective of video art—The Video Show at the Serpentine Gallery in 1974.

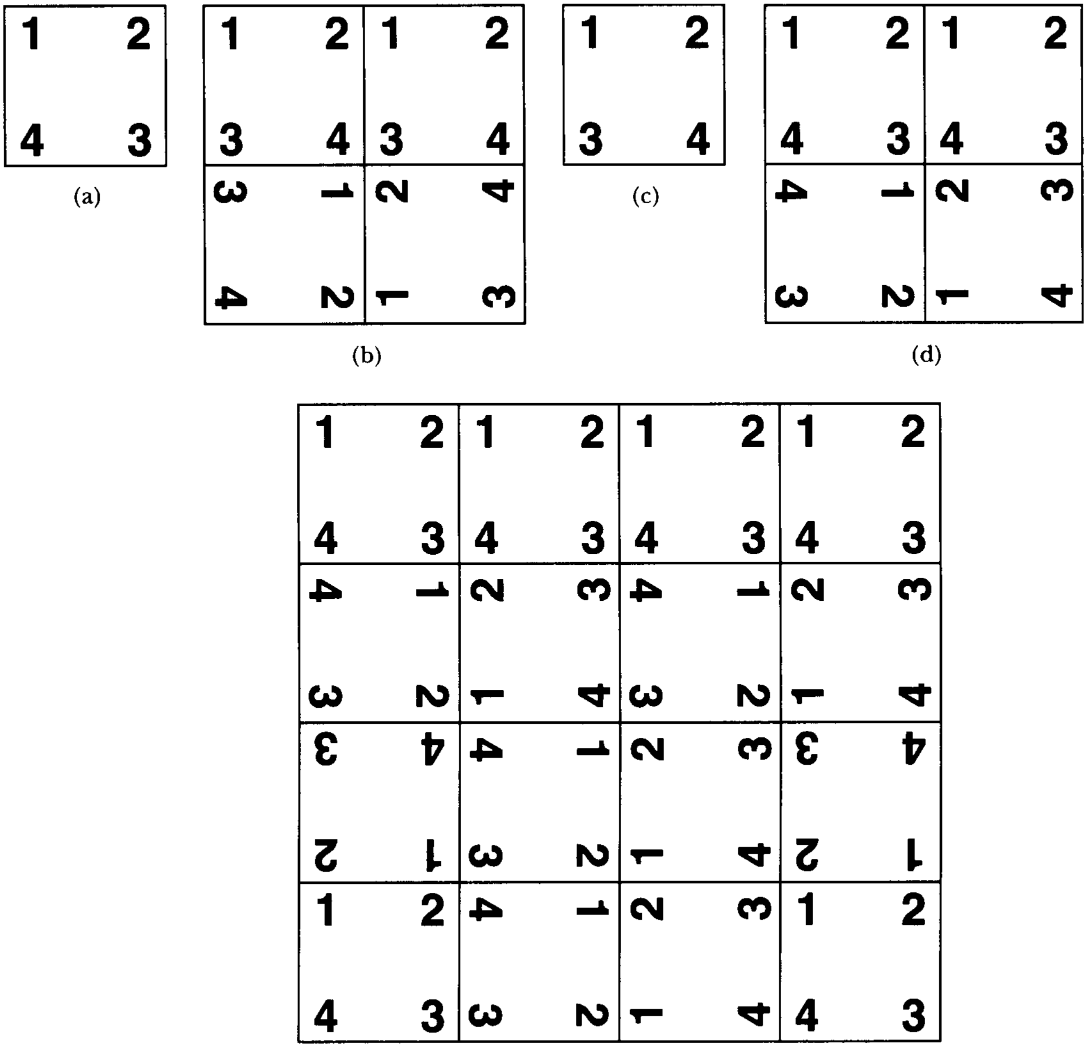
My years in the lightshow and video were a revelation. Contrary to my training, which had demanded "meaning," "significance," and "context," I discovered that a simple feedback circuit or some oil and water together with a few dies and some heat could produce large-scale immersive experiences that were, to me at least, a lot more attractive and interesting than the stuff on the walls and floors of the trendy galleries. A significant insight was my redundancy as the creator of these works. I could show someone else (who didn't need to be an "artist") how to do it. Or, possibly, I could build a machine to do it!

Being of a logical disposition I realized I needed to find a formal method of codifying and creating work of this kind as a systematic procedure. That's why the experiment with tiles that followed my reading of Ehrenzweig was such a revelation. But my interest, some might say obsession, with tiling systems has a longer history. Back in 1967, while still an undergraduate student at Manchester, I had produced a series of tile drawings (Figure 15.2) that were probably the first pieces of work that I ever made that I considered to be significant in the sense of "originality" or of being unique to me as their creator. I equate this to the idea of "personal signature" in the emotive or self-expressive arts.

Figure 15.2(a) shows a simple square tile with its corners labeled with the ordinal symbols 1–4 in clockwise order.

Figure 15.2(b) shows a two-by-two arrangement of the tile. The peripheral vertices are now labeled with the symbols 1–4 in a clockwise zigzag pattern. The internal shared central vertex repeats this pattern in a counterclockwise sense. Note the emergence of new symbol relationships at the adjacent edge vertices.

Figure 15.2(c) is a simplification of Figure 15.2(b) showing only the peripheral vertices and the zigzag pattern. The tile from Figure 15.2(c) can now be arranged in a similar fashion to that in Figure 15.2(b). The result is shown in Figure 15.2(d). Note how this restores the clockwise order of the symbols in the peripheral vertices. Also the inner shared vertex repeats this order but in a counterclockwise sense. Compare the relationship of the adjacent edge vertices with those in Figure 15.2(b). Note also that Figure 15.1(a) is a simplification of this arrangement. This demonstrates that applying the same arrangement procedure twice returns the peripheral and central vertex labels to the same state.



(a) (b) (c) (d) (e)

15.2 FIGURE An early series of tile drawings.

In Figure 15.2(e) we see the full expansion without simplification. It is clear that this process can continue indefinitely. Every second expansion will restore the initial conditions at the peripheral and center vertices while producing an expanding set of codes at the ever new intermediate vertices. It would be 10 years before I became acquainted with the work of Benoit Mandelbrot and "self-similarity," and even longer before I first heard the term "emergence" used in the sense that we understand it today. However, I'm now aware that back in 1967 I glimpsed these concepts in creating and studying these drawings.

Soon after I made these drawings I showed them to my lecturers, who dismissed them as inconsequential and irrelevant and then suggested that I reconsider my career in the visual arts. Not long after this I dropped out and began working with the lightshow.

Six years later I enrolled in the College of Art at Liverpool Polytechnic with the express intention of learning about computers. After a few months learning Fortran and the graphics package Gino-F, I began to develop a tile-based image-generating system. Although I initially used a random number generator to drive the system, I soon became dissatisfied with the simple equation of randomness with intuition. I recalled my earlier interest in "The Game of Life" and began to devise both deterministic and probabilistic CAs to create the input data for the system.

At Liverpool the painters were unsympathetic to my work and I transferred to sculpture. This department included several members of the 1960s Kinetics Group, and they were very supportive and helped me develop a small digital electronics lab. I kept my head down and took my lecturers' advice when they suggested I make some 3D stuff for my final assessment.

Then in 1977 I began two years of postgraduate studio work at the Slade School of Art at University College London. The Computing and Experimental Department had been formed in 1974 by Malcomb Hughes (then head of postgraduate) and Chris Briscoe (who had begun working with computers as an undergraduate student at Portsmouth College of Art) together with an alumni endowment that helped procure a Data General Nova II minicomputer.

The late A-Life pioneer Julian Sullivan was also on the staff, and the late Edward Ihnatowicz, who had created the early adaptive robots "SAM—Sound Activated Mobile" and "The Senster" was a regular visitor. Harold Cohen, who was then working on the early version of his drawing automaton "Aaron," visited whenever he was in the country. Darrel Viner, who had been working with the computer graphics pioneer John Vince at Middlesex Polytechnic since 1972, was also around. The place was a magnet for artists working with computers and generative systems. Many of them were involved with automata or other procedural or rule-based systems, and we were all fascinated by the area that would later be called "artificial life" or A-Life.

I also joined the Computer Arts Society, which had been founded in 1968 at Event One at the Royal College of Art. Meetings were held at the late John Lansdown's offices in Bloomsbury Square. John became a friend and mentor who, after seeing some of my work, invited me to look into the dynamic generation of unique foliage drawings for use on CAD architectural plans and perspectives. This project introduced me to Mandelbrot's work on fractal, iterative, and nonlinear systems—the area that has now been dubbed "chaos theory" (Mandelbrot 1977). John published a brief overview of this work-in-progress in his "Not Only Computing—Also Art" column (Lansdown 1978).

It's interesting to note that by the mid- to late 1970s the chaos field was well populated by scientists (who were nevertheless often working "underground" to protect their career status) and by artists. By contrast the nascent A-Life field was then almost exclusively the domain of artists.

15.5 EARLY COMPUTER WORK

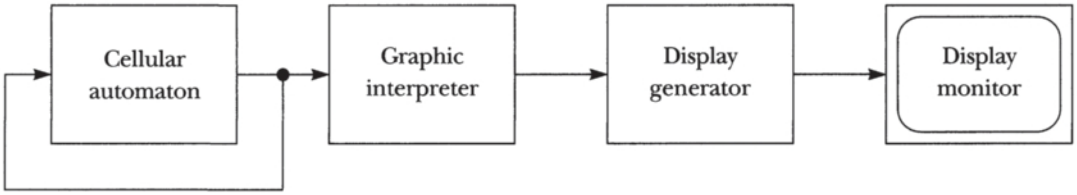
The system that I developed, then refined, first at Liverpool Poly, then at the Slade School, consisted of three components: a cellular automaton, a graphics interpreter, and a back-end display generator (Figure 15.3).

I have worked most often with simple CAs that are identical or similar to Conway's "Life." The automaton is based in a regular rectangular matrix where each cell of the matrix can have one of only two states. In general these states can be symbolized as "empty" and "occupied." Such a matrix can be represented by a one-bit array where 0 = empty and 1 = occupied. This array is the current time slice. The next time slice is calculated by applying a set of rules to each cell in the matrix. These rules examine the immediate neighbors of the cell (Figure 15.4):

1. If the cell is occupied and it has 2 or 3 neighbors occupied, then it will remain occupied in the next time slice.
2. If the cell is empty and has 3 neighbors occupied, then it will become occupied in the next time slice.
3. Otherwise the cell will be empty in the next time slice.

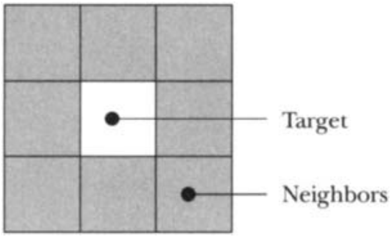
When the rules have been applied to all the cells in the matrix, the next time slice replaces the current time slice and the process is repeated. If any readers are not familiar with Conway's "Game of Life," then Poundstone offers an excellent introduction (Poundstone 1987).

Conway's rules can be implemented in a simple look-up table (Table 15.1). Since each cell has eight neighbors, the neighborhood family consists of 256 members, and we can also illustrate Conway's rules using a series of state diagrams (Figure 15.5). In each of these state diagrams, the top row and leftmost column outside the square are four-bit nybbles that address the columns and rows of the diagram. Note that rule 3 is implied by the empty spaces in Figures 15.5(b) and 15.5(c).


Cellular automaton
Graphic interpreter
Display generator
Display monitor

15.3 FIGURE

Schematic of the CA graphic system.


Target
Neighbors

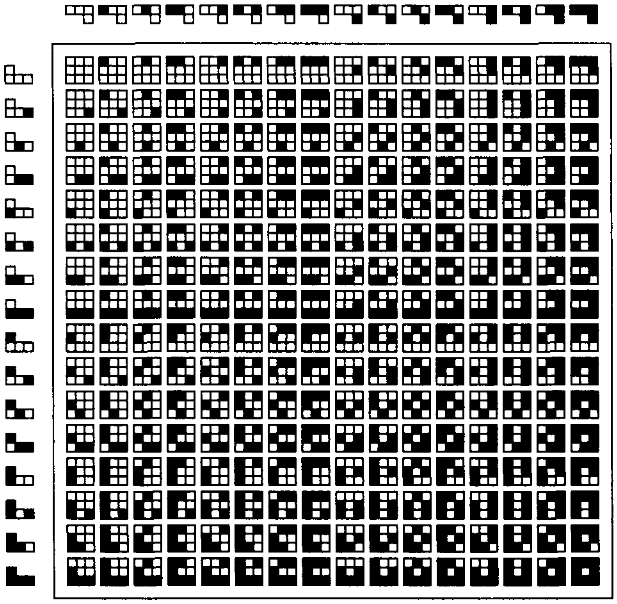
15.4 FIGURE

A target cell surrounded by its neighbors.

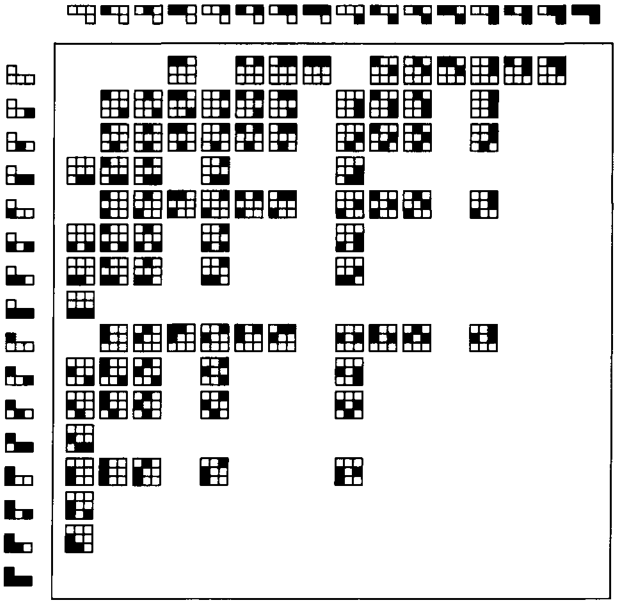
Current Time Slice Cell	Future Time Slice Cell State	
No. of Neighbors Occupied	Current Cell Is Occupied	Current Cell Is Empty
0	0	0
1	0	0
2	1	0
3	1	1
4	0	0
5	0	0
6	0	0
7	0	0
8	0	0

15.1 TABLE

Rules for Conway's "Game of Life."



(a)

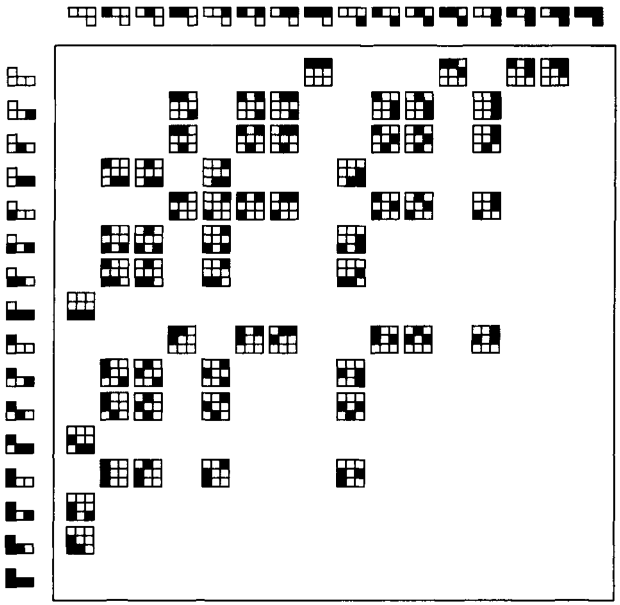


(b)

15.5

FIGURE

(a) All possible neighborhood states. (b) Neighborhood states that allow an occupied cell to remain occupied (rule 1).



(c)

(c) Neighborhood states that enable an empty cell to become occupied (rule 2).

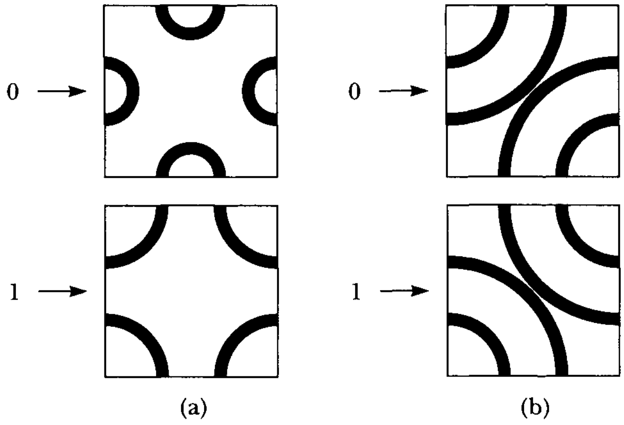
Edge conditions (where a cell does not have the requisite neighbors) are often dealt with by wrapping the array so the bottom edge is considered adjacent to the top and the left edge adjacent to the right. The finite rectangular array represented becomes equivalent to the continuous surface of a torus or doughnut. This arrangement is often referred to as "wraparound."

For every time slice the CA produces a single-bit 2D array of data as output. This array is the input to the graphics interpreter. The graphic interpreter has a fairly simple task: It controls the way that the symbolic input array maps to an actual graphic layout. The mapping is one-to-one. For each cell in the 2D bit matrix there will be a corresponding tile in an equivalent 2D graphic matrix. For single-bit input the mapping is simple (see Figure 15.6).

Most of my recent work uses multiple-bit arrays. Although I have worked a little with CAs that operate on multiple bits I have in general preferred to derive a multiple-bit solution by integrating single-bit arrays over time. For example, we can consider an identical cell in two time slices T_n and T_{n+1} (Table 15.2).

Two-bit mapping is illustrated in Figure 15.7 and three-bit in Figure 15.8.

Time integration imposes an important constraint. For example, the three-bit code 011 can only change to 110 or 111 in the next frame. The code shifts left one bit and then the CA provides the least significant bit.



15.6 FIGURE Two examples of single-bit mapping to tiles.

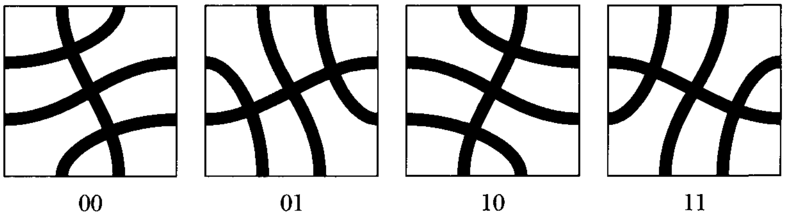
Cell State in T_n	Cell State in T_{n+1}	Anthropomorphic Interpretation
0	0	Void
0	1	Being born
1	0	Dying
1	1	Being alive

15.2 TABLE Possible states of one cell in time slices T_n and T_{n+1}.

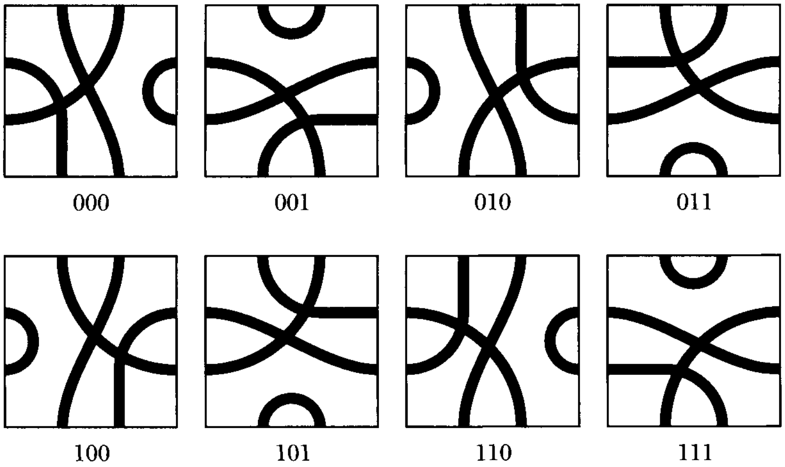
Although in my early work I often mapped bit states onto a set of different tiles (Figure 15.6(a)), I have more recently chosen to map onto families derived from rotations and mirror rotations of a single tile (Figures 15.6(b), 15.7, and 15.8).

The tiles I use have patterns on them that, in the final piece, dominate the visual appearance of the work. Many viewers are, in fact, surprised to discover the underlying tile matrix and the relative simplicity of the elements that make up often complex images. The idea of complexity emerging from simplicity—or to use the older homily, "the whole is greater than the sum of the parts"—has been a guiding concept behind my work for longer than I can remember. I find myself equally attracted to holism and reductionism and constantly oscillate between these two extremes.

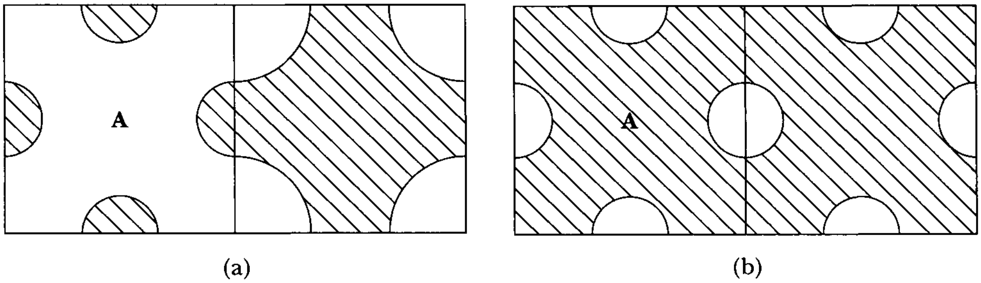
By patterning the tiles it's possible to explore ambiguities like those illustrated in Figure 15.9. The negative space has been hatched in both cases. In this way it is possible for the work to explore and/or reveal features like boundary,



15.7 FIGURE A two-bit state is mapped onto a family of four tiles.


000
001
010
011
100
101
110
111

15.8 FIGURE A three-bit state is mapped onto a family of eight tiles.



15.9 FIGURE (a) The area tagged A is most likely to be read as the positive or foreground space; (b) here it is more likely to be read as the negative or background space.

closure, inside, outside, negative, positive, foreground, background, inversion, and crossing. Readers who are familiar with Spencer-Brown's *Laws of Form* may now share my astonishment that there it has had so little direct influence on my work despite the fact that so many similar guiding concepts are shared!

It's also necessary for me at this point to issue another disclaimer! In exploring concepts or features like these I am not trying to understand them in the way that a cognitive scientist might attempt to do. Nor am I trying to create models or simulations that help us understand creative behavior or perception. As an artist I am simply exploiting such concepts, exploring them and mining them. This is not to say that I don't, at least at some level, understand them, but that such understanding is not perceived by me as being of any particular relevance to the production process of my work. It is not intended to be an illustration of such concepts; however, it's quite legitimate for me or for others to interpret the work in relationship to these concepts.

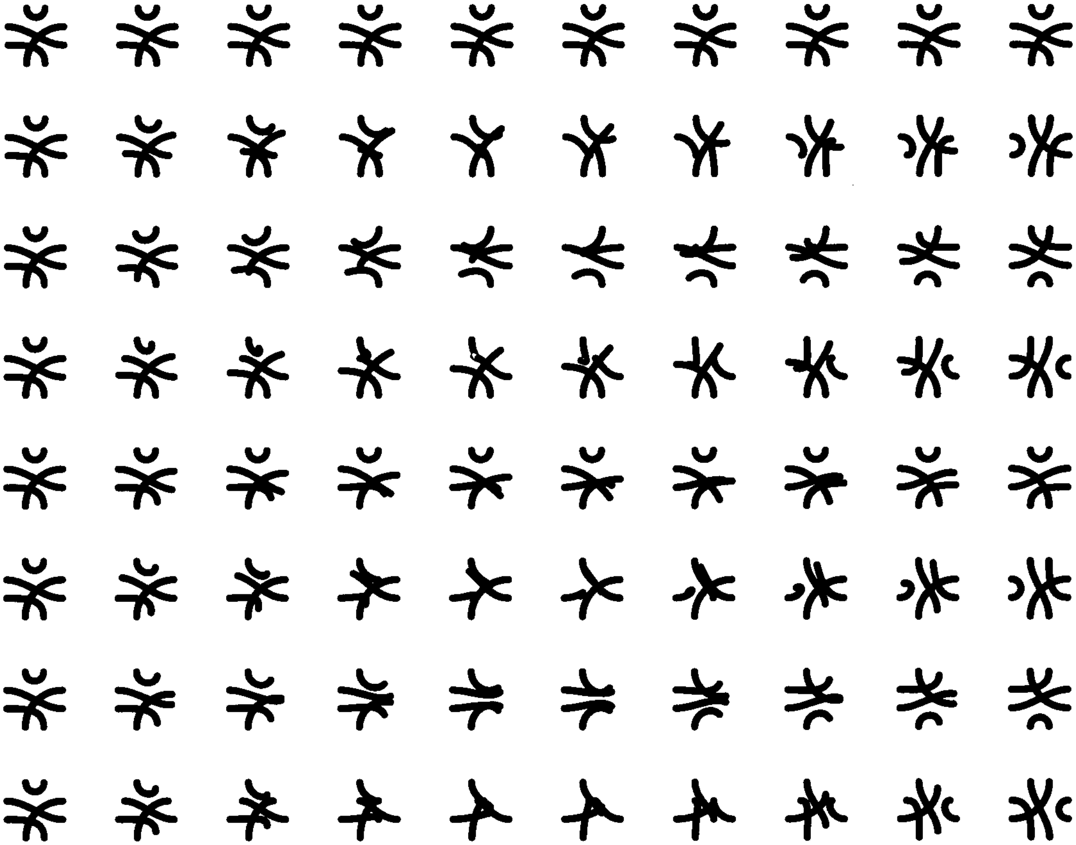
15.6 RECENT WORK

In 1986, I gave up my PC-AT and the use of mainframe and minicomputers for an Apple Macintosh, and this has been my preferred platform ever since. Around 1988 I became aware of a software application called VideoWorks, subsequently renamed Director, that has since become the de facto standard for multimedia authoring worldwide. Director V7 is a powerful object-oriented fourth-generation application generator that allows relatively naïve users to create sophisticated products for CD-ROM and the Web. For more advanced users it supports Lingo, a well-featured object-oriented programming language. Since the mid-1990s it has been possible to create and support products on both the Mac and PC systems.

All of my recent time-based works have been produced using Director. These include *Infinite Permutations V1* (a one-bit system, 1993/94), *Infinite Permutations V2* (a two-bit system, 1994/95), and *SAND LINES* (another two-bit system completed in 1998). During 1999 and 2000 I have been working on a new piece, provisionally titled *Chromos,* which will be a three-bit system.

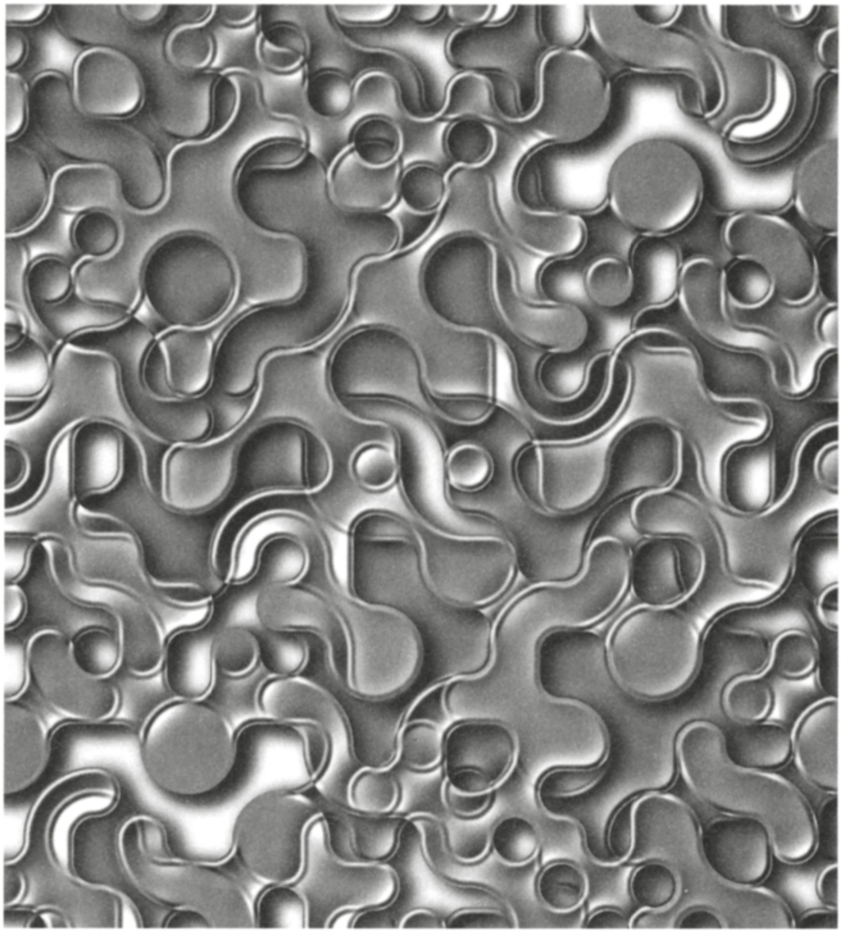
Several of these works utilize precomputed animation tables where, for example, one member of a tile family is inbetweened to each other member (Figure 15.10).

These drawings of animation tables are so interesting in themselves that I'm planning to exhibit the work in an installation that will include a large-format projection of the time-based component together with wall-hanging and book-format working drawings and other related material.



15.10 FIGURE Page 1 of the animation table for a new work *Chromos,* where family member 1 inbetweens via 10 stages to each of the eight members of the family (including itself on line 1).

In all of these time-based works I process the family of tiles using the filters and image-processing facilities of Adobe PhotoShop in order to both color and texture their appearance. Although this is sometimes merely decorative or playful, it can be used more significantly to anchor analogical or structural references that are otherwise contained in or implied by the work. For example, this mechanism can be used to both increase and decrease ambiguity or to lock into a graphic metaphor. In *Infinite Permutations V1* the color has been specifically chosen to make it difficult for the viewer to resolve the foreground/background or negative/positive conflict inherent in the pictorial representation of the work. In *SAND LINES* the image processing has been selected to consolidate the stone/sand identification and create a conflict with the animation, which does things that the materials represented could not.



15.11 FIGURE *My Gasket.* Edition of 20, 1998 Iris print on Somerset Radiant White. © 1998 Paul Brown.

In 1992, as professor of art and technology at Mississippi State University, I was able to use an Iris ink-jet printer for the first time and was amazed at its resolution, surface integrity, and color fidelity. More recently Iris, together with third-party suppliers, has introduced a variety of archival inks that can print on acid-free watercolor papers. Since 1995 I have been making limited-edition prints of images that begin as time slices of large-format tile automata like those used in the time-based pieces. I often get lost in an oceanic orgy of image processing while producing these images and occasionally even undermine or disguise the CA foundation of the work (see Figure 15.11 and Color Plates 11 and 12).

15.7 CURRENT AND FUTURE DIRECTIONS

At the time of writing I have just begun a year as artist-in-residence at the Centre for Computational Neuroscience and Robotics and the School of Cognitive and Computing Sciences at the University of Sussex. For some time now I have been following work in evolutionary computational methods and have become convinced they have an important contribution to make to my future work. The thick mists shrouding these particular stepping stones make it difficult for me to predict just how these processes may relate to my practice. I'm also concerned that prediction often prejudices the outcomes of a project and want to keep an open mind.

However, I can foresee one way that this technology could be integrated into my current system, and it's likely that this will be the focus of my work for the coming months.

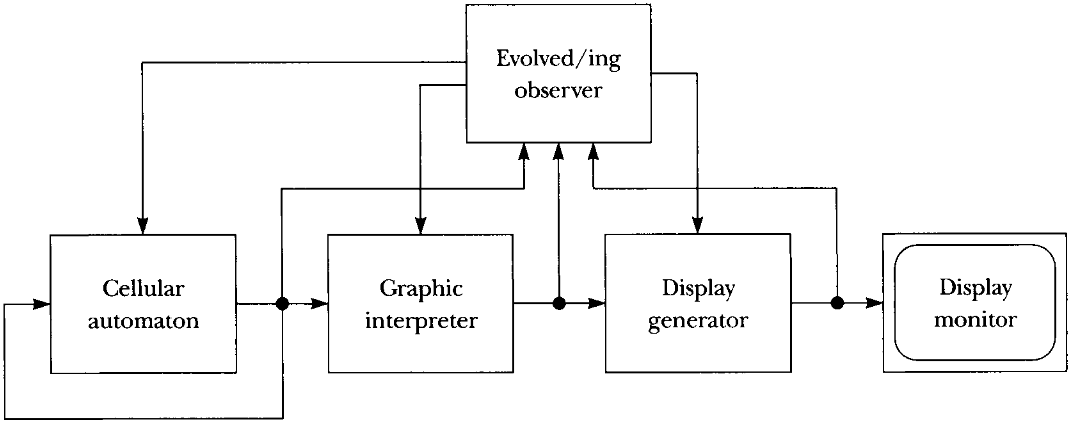
It would involve the addition of a preevolved or a dynamically evolving "observer." This observer should be able to analyze the bitmap array and/or the symbolic graphic array and/or the "actual" raster graphic display data for each frame or time slice. It would then be capable of dynamically modifying the generating CA's rules and/or the graphic interpreter's mapping rules and/or the display generator's image-processing filters.

This would be a learning system that would be capable of evolving certain kinds of graphic behaviors (Figure 15.12).

I indicated in my opening paragraphs that the far shore of my life's work is completely invisible or may not exist. However, as I have implied in the later text, I do have some long-term ambitions. The main one is to contribute to the development of autonomous creative behavior. I look forward to automata that can create artworks that a peer group of either humans or other machines will accept as legitimate creative activity.

15.8 CONCLUSIONS

The singular and remarkable success of artists like Harold Cohen is his achievement in externalizing his own personal creative drawing behavior. Aaron, the automaton he has created, produces 100% genuine Harold Cohen drawings. I believe that we will eventually be able to create automata that will make artworks that do not bear the signature of the creator of the system—that the system will be capable of evolving its own personal style.



15.12
FIGURE
Proposed schematic for an evolving time-based system. Compare this with Figure 15.3.

I leave the reader to ponder the problem of just what the words "own" and "personal" might mean in this context.

My own feelings are that the growing realm popularly referred to as cyberspace is an ecosystem and that creatures must evolve to exploit that space. Since a primary fitness characteristic would be to hide from humans (who would almost certainly try to destroy them), they may already exist! Such entities would have access to a vast repository of information from both real-time sources and from archives. I find it inconceivable that creatures with such a broad bandwidth of input "sensations" will not develop behaviors that are analogical to human art making.

ACKNOWLEDGMENTS

I would like to thank Dr. Phil Husbands, joint coordinator of the Centre for Computational Neuroscience and Robotics, and Richard Coates, dean of the School of Cognitive and Computing Sciences at the University of Sussex, for the invitation to join their program for a year as artist-in-residence.

In particular I must thank Gavin Sade of the Communication Design program of the Academy of the Arts, Queensland University of Technology, who recently turned my hacked Lingo code into a series of elegant modular objects.

I am especially grateful to the Australian Commonwealth Government and the Australia Council, its arts funding, and advisory body for their award of a New Media Art's Fellowship, which will fund my work throughout 2000 and 2001.

This chapter is based on a presentation I made at First Iteration, a conference arranged by Jon McCormack and Alan Dorin at Monash University, Melbourne, December 1–3, 1999 (Dorin and McCormack 1999).

REFERENCES

Biederman, C. (1948). *Art as the Evolution of Visual Knowledge.* Red Wing.

Dorin, A., and J. McCormack (eds.) (1999). *Proceedings of the First Iteration Conference.* Centre for Electronic Media Art, Monash University.

Ehrenzweig, A. (1965). *The Psycho-analysis of Artistic Vision and Hearing: An Introduction to a Theory of Unconscious Perception.* Second edition. Braziller.

Ehrenzweig, A. (1967). *The Hidden Order of Art: A Study in the Psychology of Artistic Imagination.* Weidenfeld.

Gardner, M. (1970). Mathematical Recreations. *Scientific American* 223(4):120–123.

Kay, A. (1984). Computer Software. *Scientific American* 251(3):41–47.

Lansdown, J. (1978). Only God Can Make a Tree. In *Not Only Computing—Also Art,* Computer Bulletin, British Computer Society (September).

Mandelbrot, B. B. (1977). *Fractals: Form, Chance and Dimension.* Freeman.

Poundstone, W. (1987). *The Recursive Universe: Cosmic Complexity and the Limits of Scientific Knowledge.* Oxford University Press.

Reichardt, J. (1969). Institute of Contemporary Arts. *Cybernetic Serendipity: The Computer and the Arts.* Special issue of *Studio International.*

Spencer-Brown, G. (1969). *Laws of Form.* George Allen and Unwin.

Wilhelm, R. (ed., trans. German), and C. F. Baynes, (trans. English) (1923/67). *The I Ching or Book of Changes.* The Richard Wilhelm translation (1923) rendered into English by Cary F. Baynes, Princeton University Press, third edition (1967).

This Page Intentionally Left Blank

16 Evolutionary Generation of Faces

CHAPTER

Peter J. B. Hancock University of Stirling
Charlie D. Frowd University of Stirling

16.1 INTRODUCTION

This chapter describes a system that allows the evolutionary generation of near-photographic quality face images. The major anticipated use for such a system would be for allowing eyewitnesses to a crime to construct a likeness of the suspect. Current methods for doing this center on photofit-like composite systems. The user is presented with a large range of possible alternatives for major features such as nose, mouth, and eyes and attempts to reconstruct a face piecemeal. The original photofit does not work very well (Ellis, Davies, and Shepherd 1978), probably precisely because it does break the face into components in a way that we do not usually do when viewing a face. More recent systems, such as CD-fit, are far more flexible, allowing the position, size, and shape of individual features to be altered interactively on a computer. However, the essential problem of being a composite system remains. If you change, for example, the nose, our perception of the eyes and mouth may well change: we perceive faces holistically. The resultant images also look rather synthetic. The system described here produces face images by recombining principal components (eigenfaces), which are inherently global in nature and produce more photographic-like images.

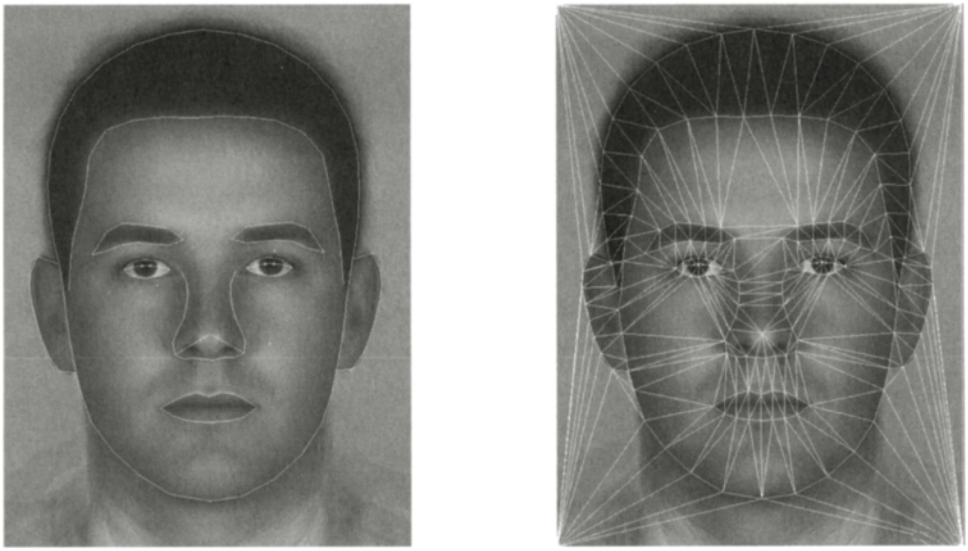
16.1.1 Eigenfaces

Principal components analysis (PCA) is a well-known statistical technique for reducing the volume of a set of data. It pulls out mutually orthogonal axes that account for the most variance in a data set. Applied to a set of face images, it will produce ghostlike images known as eigenfaces. These eigenfaces form a basis set for the set of faces that were analyzed: By recombining eigenfaces in appropriate amounts, the original faces can be reconstructed. Because the eigenfaces are ordered in terms of the amount of variance they account for, it is usually possible to use, say, only the first 20 out of a set of 50 and produce quite good reconstructions of the original faces. The 20 component values provide a very compact description of the face. This forms the basis for PCA-based facial identification: A novel face is projected through the eigenfaces to obtain component values that may easily be compared with those in the data set (Turk and Pentland 1991).

If eigenfaces are recombined randomly, then a facelike image not necessarily resembling any of those in the original set may be generated. This appears to offer a means of producing novel faces in a holistic fashion (O'Toole and Thompson 1993). Unfortunately the images produced tend to be very blurred, because of the variations in face shape. Typically the face images are aligned by the center of the eyes prior to PCA. Variations in shape mean that other features, such as nose and mouth and the outline of the face, will appear in different locations. These variations are captured by the PCA, so that when components are recombined, everything but the eyes is blurred.

This problem may be addressed by locating the positions of all the key facial features and the outline and then morphing each face to an average shape prior to PCA (Craw and Cameron 1991). Our current face model defines 173 points around the face, as in Figure 16.1 (it is not an even number because of the one in the middle of the nose!). In order to achieve some consistency in the location of this many points, the model is constrained. For example, the points marking the corners of the mouth are free to move wherever the operator wishes. Having located them, the seven points that define each lip are fixed to be equally spaced horizontally and are thus free to move only vertically. The triangles used for morphing the images are shown in Figure 16.1(b). They are generated by a public-domain Delaunay triangulation program, which may not be optimal in terms of producing an aesthetically pleasing final image. We now think that the face model may require some modification to add extra points, around the eyes, nose, and the edge of the image. Many of the border triangles are very elongated, which can cause distortions of the ear and hair fringes.

Despite some problems, the model produces a very acceptable alignment of the main facial features, as Figure 16.1 indicates. With all the features appearing in essentially the same locations, the image PCA is left to account for more subtle variations in the surface appearance. Recombining the resulting eigenfaces produces much sharper images, but all of the same, average shape. Figure 16.2 shows an average female image, produced using a rather simpler face model with fewer reference points, and illustrates the effects of the first four eigenfaces



(a) (b)

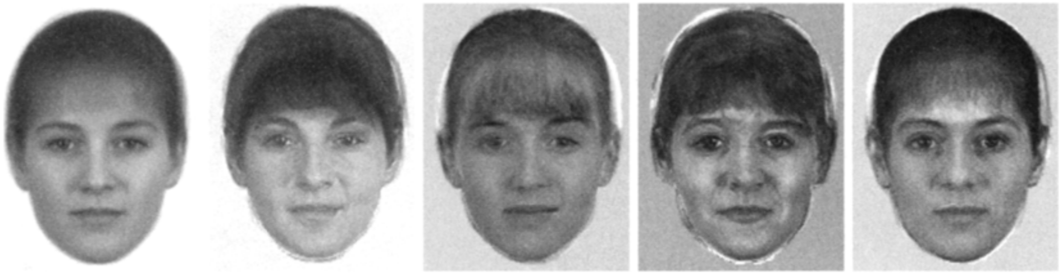
16.1

FIGURE

An average male face (a) with key features marked and (b) with the morphing triangles.

from a set of 20 faces. Although there are some fringing effects around the edges, caused by the more limited face model employed, the resultant images are quite sharp. Each of the four eigenfaces has produced a distinctly different appearance for the whole face, illustrating the holistic nature of the PCA coding scheme.

These faces, while visibly different, are all the same, average shape. This can be overcome by performing PCA also on the shape vectors to produce the principal components of facial shape variation, *eigenshapes*. The effect of these eigenshapes can be seen by applying them to the key points of the average shape-free face. These variations are shown in Figure 16.3. The degree of variation has deliberately been exaggerated in order to illustrate the effects clearly. In this case, the first component codes overall head size. With no point of reference to estimate size, such variation is of little consequence for our purposes, so we exclude this component from the reconstruction. The second component is coding the width of the head, clearly an important variable. The third and fourth components are mostly coding for head angle—tilt and rotation, both of which are artifacts caused by the difficulty of ensuring that people hold their head straight



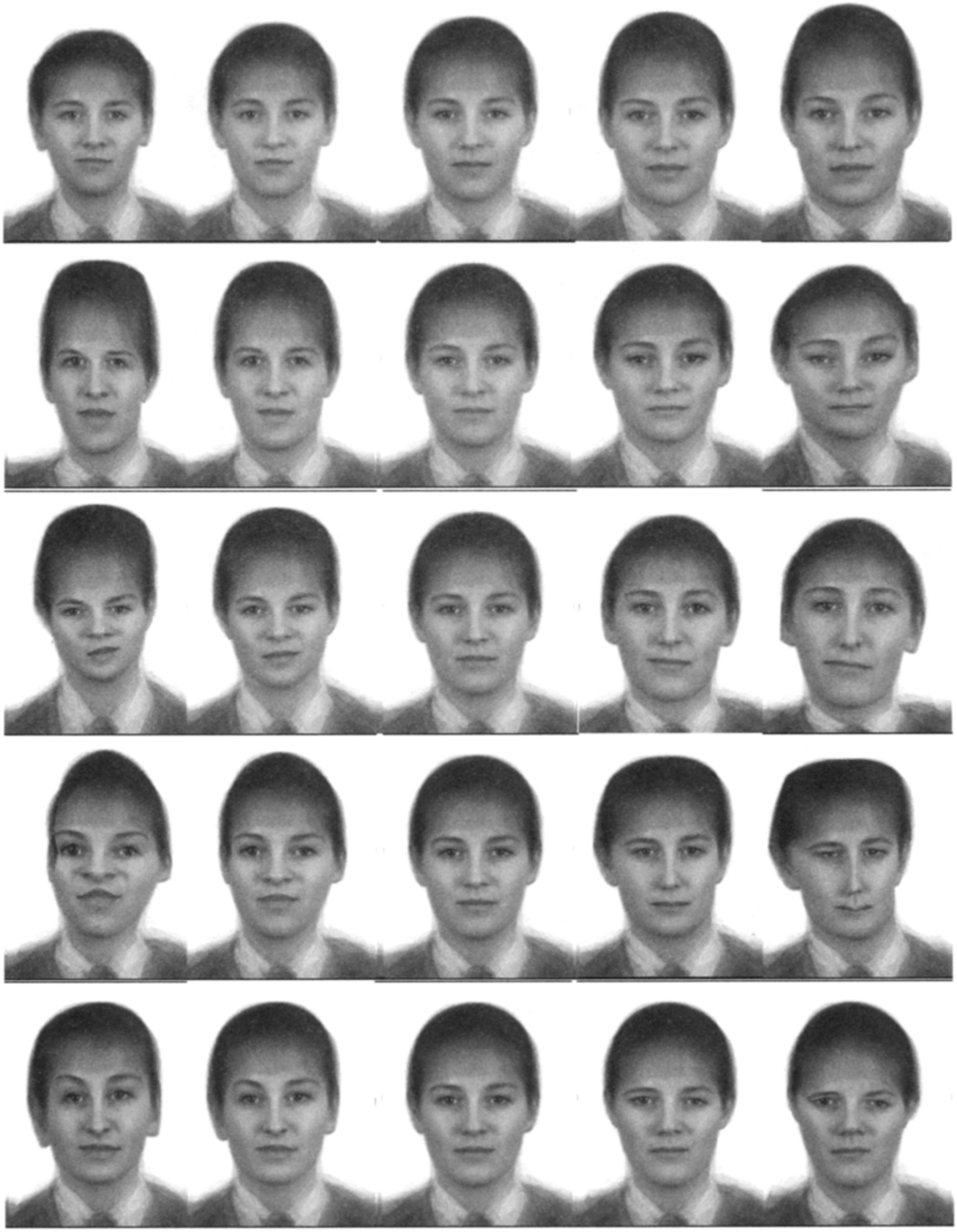
16.2 FIGURE An average female face (left) and the effect of adding the first four eigenfaces to it.

when being photographed. Often another component will code for the remaining degree of freedom in head movement, nodding. Ideally, we would identify and remove all three orientation components from the reconstruction set, since their only effect is to distort the image.

These eigenshapes can also be recombined randomly to produce a novel face shape. The shape-free random face image is converted to its new shape by morphing. Figure 16.4 shows an example of a randomly generated face produced by the low-resolution pilot system.

16.1.2 Evolutionary Face Generator System

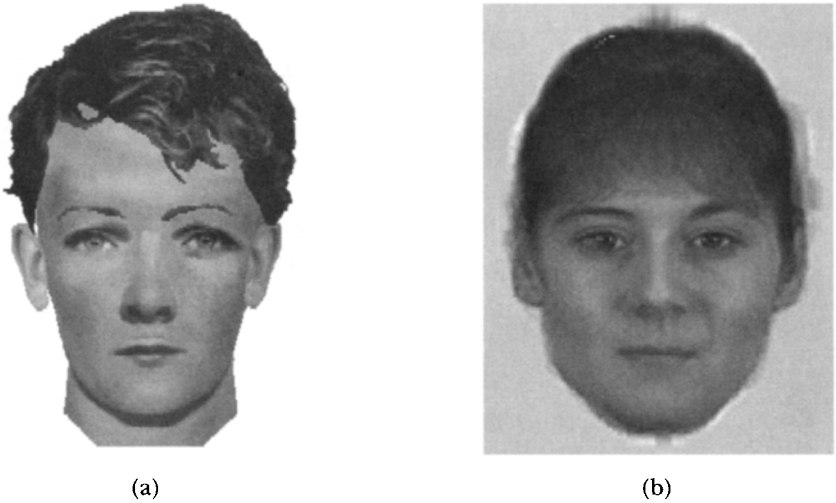
We thus have the basis for a system that can generate near-photographic quality faces at random, at least within the space of faces accounted for by the images used to generate the components. The question then arises of how to let users interact with the system to try to re-create a target face. One possibility would be to provide a set of sliders that vary the amount of each component interactively as in O'Toole and Thompson (1993). This would be computationally intensive, requiring specialist hardware to perform in real time. It might also be rather confusing, with many sliders and little obvious sense to them. While it has been shown that PCA does show significant correlations with human perceptions of faces (Hancock, Burton, and Bruce 1996), many components produce changes that are rather hard to understand. We have therefore built an evolutionary front end, which allows users to rate each face according to likeness to the target. An underlying evolutionary algorithm uses these ratings to produce new faces, by recombination and mutation of the parameters. An evolutionary face generation system was described by Caldwell and Johnston (1991): Their system used photofit-like composites as the underlying representation. One problem with



16.3

FIGURE

Illustration of the effects of the first five eigenshape vectors.



(a) (b)

16.4 FIGURE (a) An example CD-fit image and (b) an example from the PCA-based evolutionary system.

this form of representation for an evolutionary system is that there is no sense of a continuum in the feature space: nose No. 12 is not necessarily more like nose No. 13 than it is like nose No. 14. Imagining the task of trying to impose this kind of ordering will give additional insight as to the inadequacy of this kind of composite representation of faces. Our underlying representation is a set of real numbers, which define how much of each eigenface and eigenshape to use in the construction of a face.

Initially, a pilot system was constructed to test the general feasibility of the approach. This system uses a database of 20 young adult women, all with short or tied-back hair. Variations in hair present a problem, discussed further below. The generation system used 10 image and 10 shape components, giving a genetic string of 20 real numbers. The initial population was generated using Gaussian random numbers, with a standard deviation proportional to that of the matching principal component, so as to produce images within the normal range. Users rated each face on a scale from 0 to 10; this number was then used to perform fitness proportional selection. Mutation was applied using Gaussian random variables, again with appropriate variance for the component in question, while crossover was uniform (i.e., 50% probability of a cross between each component variable), since we have no particular reason to think that there will be linkage between nearby components. An example of the kind of image generated by the system is shown in Figure 16.4, with a random image from CD-fit for comparison.

16.2 TESTING

While it is evident that the system is capable of producing reasonably high-quality images, it is not yet clear how useful it will be for producing a likeness of a target. One issue is simply that of coverage: The components only cover the space of the images that are analyzed (at best), and the system cannot produce faces of a different age or race without additional example images. This may not be a problem, since initial questioning of the witness should identify the correct target group. More fundamental is (1) whether an evolutionary system will, in practice, allow suspects to navigate face space successfully and (2) whether principal components are a good underlying representation.

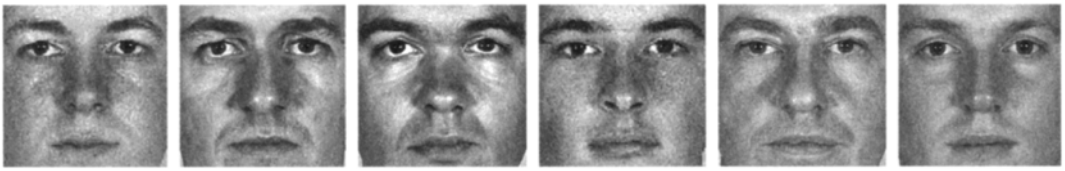
We have commenced our formal evaluation of the system with a simplified version, recoded to run under Windows NT, better to match likely end users if the system is successful. One simplification is the omission of shape variation, which removes the requirement for on-the-fly morphing of the images. We also cropped tightly into the face, to remove external features such as hair and ears. Our initial concerns were (1) whether users would be able to use the system to create a likeness of a target when it is available for comparison and (2) whether it would be better to display one face at a time for matching, or several at once.

16.2.1 Apparatus

A Pentium PII PC running at 350 MHz was used to run the experiment. Faces were displayed on an Iiyama 21-inch monitor. The PCA model was derived from 35 full-face Caucasian males (extracted at 8-bit monochrome with a resolution of 300 × 400 pixels in BMP format).

16.2.2 Generation of Face Images

The 35 male faces were put through a shape normalizing process (morphing) that aligned facial features. Key locations around eyes, ears, mouth, and so on and the outline of the head are located manually for each face. The average location for each point is calculated and each face morphed to this uniform shape. This stage is necessary to avoid feature misalignment when faces are generated randomly later on. Following this, each face was cropped to 176 × 171 pixels so that just the inner features of the faces remained, serving to limit further misalignment difficulties caused particularly by varying collar lines and hairstyles.



16.5 FIGURE The first six cropped images used to generate the eigenfaces.

Figure 16.5 shows the first six shape normalized faces cropped to the appropriate dimensions.

Principal components were extracted from these face images, and the top 18, which explained the most variance, used as the basis set for generating new, random faces. To produce a new face, 18 floating-point random numbers (drawn from a uniform distribution with mean 0 and range -1.0 to $+1.0$) were generated and scaled, first by the standard deviation of the corresponding eigenface (to maintain an appropriate influence of each coefficient) and then by a factor of 35 (to produce sufficient variability of facial features). This procedure was repeated until 50 faces had been produced. Following this, a process of nearest-neighbor pruning was applied until 12 faces remained.[1] Seven of these were selected as the target set in the experiment.[2] Each face was normalized for brightness[3] and intensity[4] to avoid gross differences in image lighting effects. The first five targets are shown in Figure 16.6.

A further six faces were randomly generated and normalized for brightness and contrast. These images were used as the initial population and were presented to participants (via a computer monitor) at the same time as the first target face. In condition A, the user saw the target and just one of the faces in the population, in condition B all six faces in the population were presented for rating at the same time. A participant was assigned to one of these two groups.

16.2.3 Evolutionary Algorithm

In both conditions, following the rating of the six randomly generated faces, a new population of faces was created using an evolutionary algorithm. A roulette

---

[1] Nearest-neighbor pruning randomly selected a face and then discarded the one from the remaining images that had the lowest mean-squared error. This was repeated until only 12 faces remained, resulting in a set of highly dissimilar random target faces.

[2] The rest were put aside for future work.

[3] A midrange brightness level (128) appeared appropriate.

[4] The contrast level is equal to the standard deviation of an image. A standard deviation of 25 was found to give good results for a number of faces.

16.6 FIGURE The five randomly generated target faces.

wheel mechanism based on the six unscaled rating scores selected a pair of "parent" faces. The 18 parameters are held as real numbers and recombination operates between these, not within them as would be the case in a bit-string GA. Uniform crossover thus randomly selects parameters from the coefficients of the parent faces. A small probability of mutation was applied such that 1 in 20 coefficients was replaced with an appropriate random value.[5] The resulting face was once again scaled for brightness and contrast. The procedure was repeated a further five times to generate a total of six new faces. These faces were then presented to the user for rating. A total of 11 cycles of the evolutionary generator was run for each of the five target faces for each participant.

All participants were given a short practice session involving a single evolutionary cycle for the first two of the seven target faces. Rating scores, population faces (eigenface coefficients and actual image files), and demographic information (age and gender) were among the data collected for each participant; data from the practice session was discarded.

16.2.4 Participants

Eighteen students at the University of Stirling participated in the experiment. They were paid at the rate of £5/hour. There were 11 females and 7 males; 9 were assigned to condition A and 9 to condition B. Their ages ranged from 18 to 26, with a mean of 21.

16.3 RESULTS

One issue is how to assess the extent to which a face generated by a user matches the target. Ideally, generated faces would be shown to other observers who could either rate them for likeness, or perform a same/different matching task. For our initial testing we wanted something faster, and the mean-squared error (MSE) was used as the main measure of performance. It was computed as the average of the sum of the squared difference between corresponding pixels in a target face and an image in the face population. Since each face is normalized for brightness and contrast, remaining differences in MSE may be expected to be perceptually salient. This was confirmed by generating a number of faces differing by known MSE from a target and asking colleagues to place them in order of likeness. Their order generally agreed with the "correct" order, as defined by MSE.

---

[5] Scaled by both the standard deviation of the corresponding eigenface and a factor of 60.

The performance in terms of MSE was obtained for each participant. An example (participant 3) is shown in Figure 16.7 for target 1 against each image in the face population; for clarity, other evolutionary generations are not shown.

Generation 1 shows the six MSE scores compared with target 1 at the initial presentation of the face population; the average MSE is 355 (s.d. of 113). By generation 5 (four cycles of the evolutionary generator), the MSE dropped substantially to 193 (s.d. of 51); this reduces further to 90 (s.d. of 29) by the last generation. A within-subjects two-tailed t-test reveals a significant decrease in MSE between generations 1 and 5 ($p = 0.005$), and also between 5 and 12 ($p = 0.002$).

Figure 16.8 shows the average MSE for the six faces in each generation and reveals a general decrease across generations (though some noise is present especially around generation 10).

Participant 3's MSE reduced by 125 when averaged over all five targets, a significant reduction ($p < 0.05$). In total, 10 of the participants exhibited a significant reduction in average MSE. Closer examination of the data revealed that two participants appeared to use the rating scale backward (putting a lower score for images *more* similar to the target) and six showed little correlation with rating scores and target error.

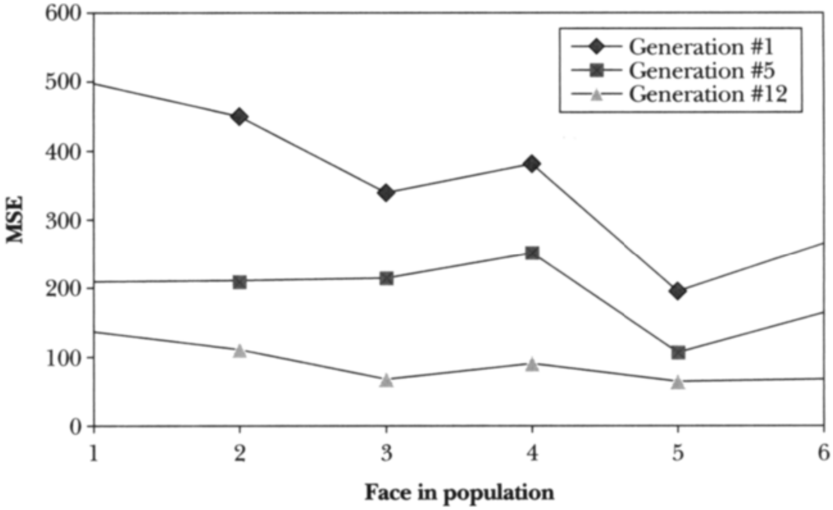
To consider general performance, a measure of cumulative reduction in average MSE for each generation across all participants and targets was used.

Figure 16.9 indicates that there was a large reduction in target error for the first few generations, but the error reduction then asymptotes to a value of 58. An exponential function[6] was found to describe more than 97% of the variance in the graph.

In terms of performance on condition A (one face at once) and B (six faces at once), it was found that group A had an initial mean MSE of 342 (s.d. of 100) and final MSE of 295 (s.d. of 55). Similarly, the initial mean MSE of B was 343 (101) and final mean was 299 (53). A within-subjects two-tailed t-test found a

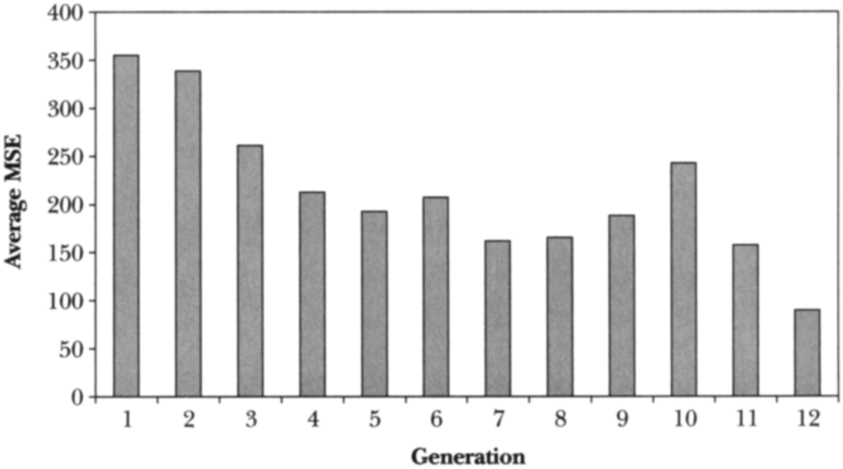
---

[6] $70 * (1 - \exp(-0.4x)) - 10$



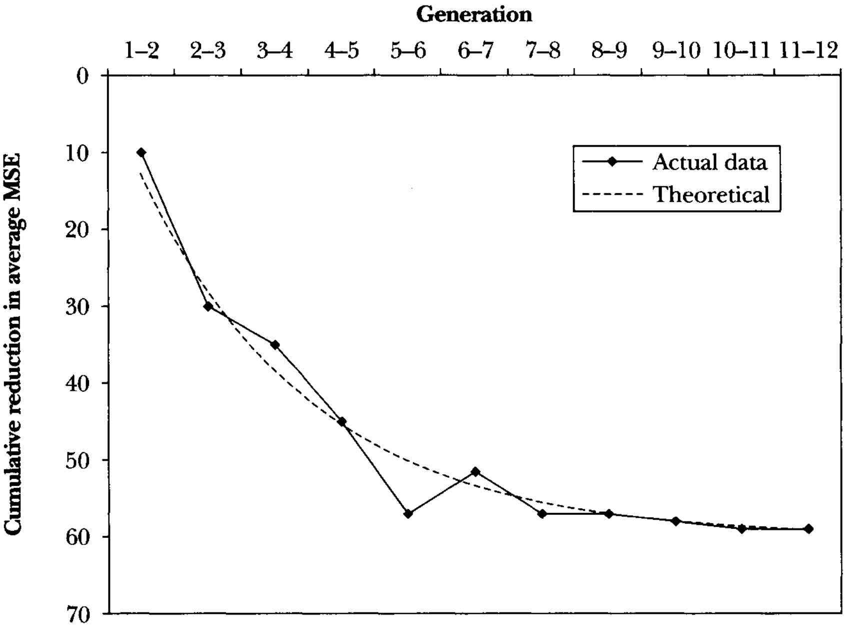
16.7
FIGURE

Performance of participant 3.



16.8
FIGURE

Overall performance for participant 3 on each generation for target 1.



16.9 FIGURE Cumulative reduction in average MSE (all participants and targets).

significant decrease between initial and final average MSE in condition A ($p < 0.001$) and condition B ($p < 0.001$). There was no significant difference between groups ($p = 0.87$). The average rating scores for the first generation was 7.1 for condition A and 6.5 for condition B. A within-subjects two-tailed t-test found no significant difference in average rating scores between the first and last generation ($p = 0.12$).

The use of average MSE seems an appropriate measure when referring to evolutionary processes since it provides a measure of average fitness of the population being evolved. Nevertheless, perhaps a more practical measure is to take the *best* MSE score from generations 2 to 12 for a specific target (since the ultimate purpose of the system is to stop when a target face has been found). If a cutoff level of 125 is chosen as a reasonable first approximation of system success,[7] 24 cases were found, representing a system success rate of 30% (excluding the two participants who incorrectly used the rating scales). No correlation was found between the best MSE of the initial population and best MSE of the following 11 generations ($r = 0.03$), so a successful outcome is not simply dependent on a fortuitous starting position.

---

[7] It has been observed that an MSE of 125 results in images that are very similar to a target. It is believed that this level of MSE could be taken as system convergence for a target. Further experiments are planned to quantify this more objectively.

16.4 DISCUSSION

It was very satisfying to observe the result from participant 3 where a face was found that reached the specified criterion in four out of five of the targets (the fifth target face only narrowly missed the criterion set). This, coupled with the observation that 30% of the attempts reached criterion, suggests that a system such as the one employed here could be successful in converging to a target image. More work is still necessary of course. Two of the participants experienced difficulties with the use of the rating scales, suggesting that caution should be used with system training. Further analysis is necessary to understand why several participants produced poor correlation scores between face ratings and MSE.

It was also pleasing to see the average MSE followed an asymptotic curve with increasing generation and that a characteristic function could be found to model average performance. The function appears to provide a method for predicting the appropriate number of evolutionary cycles necessary to reduce most of the average error. Further experiments in progress, looking at the effects of mutation rate, are being stopped after only six generations, since most of the improvement has occurred by then.

One of our main questions was whether there would be a difference between rating faces individually (condition A) and rating faces in the presence of other faces (condition B). The current results show no sign of a difference. Somewhat to our surprise, there is not even a difference in the rating scores given to faces: We had expected that the presence of all faces during rating might act as a bias input resulting in a difference in average rating scores. This would in turn affect selection pressures in the genetic algorithm and consequently performance.

The one-at-a time method (A) does not differ in underlying mechanism; it is not a steady-state evolutionary algorithm, but simply presents the six faces one at a time, then generates a new batch of six, just as the six-at-a-time version does. There can be some advantage in steady-state generation of new parameter sets for testing, in that a particularly good set becomes available at once for recombination, rather than having to wait until the end of the generation. A change to a fully steady-state algorithm might improve our performance, though with such a small population size, the difference is not likely to be large.

Another issue is how the faces should be rated. At the moment we are using a scale from 1 to 10, with a "quick and dirty" proportional fitness selection algorithm, not ideal (Hancock 1997). It seems clear that this should be replaced by a ranking scheme, in order to remove the scaling problems associated with participants' erratic use of the rating scale, but we suspect that we may do better by asking users to pick the best three, and then allowing each of those faces two reproductive opportunities in the next generation, akin to the (μ,λ) approach of evolution strategies. We think this may be an easier task for users.

One reason for the relatively poor performance of our participants here may be simply that the faces are all so similar. With no shape information and only internal features, differences are slight and may be close to the limit of what people can differentiate. We therefore intend to add in shape variation for our next tests.

The population of six is small by evolutionary standards. We have reason to believe that simple selection errors account for some of the poorer results, with lower-rated faces being inappropriately overselected and good faces getting lost in an early generations. A shift to a (μ,λ) type selection algorithm should help prevent this. We are not sure about using a bigger population, suspecting that too large a variety of faces may simply confuse a user. It is another parameter to be investigated.

The target faces for our initial trials were generated by random recombination of the components, so they are clearly within the capability of the system to reproduce exactly. Another on the list of things to test is what happens when the target is a face external to the system, and therefore potentially not well covered by the components.

The faces in this trial omitted hair. While this can be justified on the grounds that it is the most variable and easily disguised aspect of someone's appearance, it is still something that we would like to address. However, we suspect that PCA is not very suitable because of the wide variations, and that some kind of composite approach may be better for this aspect of facial appearance.

16.5 CONCLUSIONS

We have demonstrated a system capable of producing near-photographic quality images of faces. An initial version that presents only the inner features of faces, with no variation in gross shape, has been shown to allow users successfully to produce a good likeness of a target face. There remain many technical issues to consider, including selection methods, population size, and the effects of varying face shapes.

There are also significant psychological issues. Our memories are quite fragile, open to suggestion and interference. Our experiments with the system so far have allowed users to match a target image. In real life, the system will be used to try to recover a likeness from memory. We aren't sure whether presenting a user with a large number of faces to be rated in this way will simply confuse a witness. Even if this does turn out to be the case, this evolutionary methodology has the potential to be a useful tool in exploring the space of faces. If the underlying representation matches the way we code faces, then it should be relatively easy to create a desired face. Although it has some correlation with psychological perceptions (Hancock, Burton, and Bruce 1996), we do not think PCA is necessarily the best representation to use and we intend to explore alternatives.

ACKNOWLEDGMENTS

This work is funded by EPSRC grant No. GR/L88627.

REFERENCES

Caldwell, C., and V. S. Johnston (1991). Tracking a Criminal Suspect through "Face Space" with a Genetic Algorithm. In R. K. Belew and L. B. Booker (eds.), *Proc. Fourth Int. Conf. on Genetic Algorithms,* Morgan Kaufmann, pp. 416–421.

Craw, I., and P. Cameron (1991). Parameterising Images for Recognition and Reconstruction. In P. Mowforth (ed.), *Proc. British Machine Vision Conference,* Springer-Verlag, pp. 367–370.

Ellis, H. D., G. M. Davies, and J. W. Shepherd (1978). A Critical Examination of the Photofit System for Recalling Faces. *Ergonomics* 21:297–307.

Hancock, P. J. B. (1997). A Comparison of Selection Mechanisms. In T. Bäck, D. Fogel, and Z. Michalewicz (eds.), *Handbook of Evolutionary Computation,* Oxford University Press.

Hancock, P. J. B., A. M. Burton, and V. Bruce (1996). Face Processing: Human Perception and Principal Components Analysis. *Memory and Cognition* 24:26–40.

Johnston, V. S., and M. Franklin (1993). Is Beauty in the Eye of the Beholder? *Ethology and Sociobiology* 14:183–199.

O'Toole, A. J., and J. L. Thompson (1993). An X Windows Tool for Synthesizing Face Images from Eigenvectors. *Behavior Research Methods, Instruments and Computers* 25:41–47.

Turk, M., and A. Pentland (1991). Eigenfaces for Recognition. *Journal of Cognitive Neuroscience* 3:71–86.

This Page Intentionally Left Blank

17 CHAPTER

The Escher Evolver: Evolution to the People

A. E. Eiben Free University Amsterdam
R. Nabuurs Leiden Institute of Advanced Computer Science
I. Booij Leiden Institute of Advanced Computer Science

17.1 INTRODUCTION

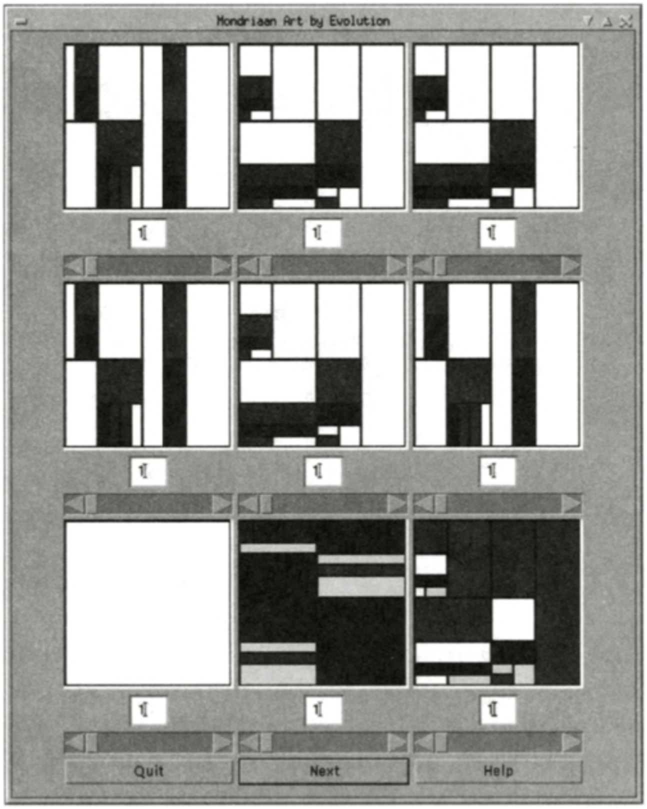
In this chapter we describe the Escher Evolver project. The objective of this project was to design and implement an evolutionary algorithm framework supporting the creation of images in the style of M. C. Escher based on subjective selection. The entire software environment has been deployed in the City Museum of the Hague as part of an Escher exhibition that ran from May to October 2000. In this chapter we will look at the technical aspects of the implementation as well as the cognitive processes regarding the visitors of the exhibition.

The Escher Evolver project had two origins. The first of these was the intention of the City Museum (Gemeentemuseum) in the Hague, Netherlands, to organize a large exhibition devoted to the Dutch artist M. C. Escher. Although Escher had no background in mathematics, his insights and self-study led him to an art form that is based on the geometry of two- and three-dimensional spaces. Particularly popular Escher works are his pictures based on tiling in the two-dimensional plane (Escher and Bool 1986; Schattschneider 1990). These images feature animal-like figures that complement each other, thus forming a complete coverage of the plane.

The second origin of this project lies in the first author's regular university course on evolutionary computation. In 1998, the programming assignment for this course was the implementation of an evolutionary system that creates images in the style of the Dutch artist P. Mondrian,[1] which allows the user to steer

---

[1] See the Mondrian resources Web page maintained by P. Montford at *http://www.aukword.demon.co.uk/piran/mondrian/resources.html.*



17.1 FIGURE Screenshot of an online Mondrian Evolver (available at *http://www.cs.vu.nl/ci/Mondriaan/Mondriaan.html*).

evolution by performing subjective selection on a given population of images. The first version of such a "Mondrian Evolver" was implemented by van Hemert and Eiben (1999), and the most recent version available through the Web relies on the contribution of B. Craenen.[2] Figure 17.1 shows a screenshot from the online Mondrian Evolver, illustrating the GUI that presents a population of nine images to the user for subjective evaluation.

Through informal contacts between the university and the museum, the idea emerged that the traditional Escher exhibition should be extended by the inclusion of a "virtual" part: computer-generated images shown to visitors on

---

[2] See the Evolutionary Art Web page maintained at *http://www.cs.vu.nl/ci/EvoArt.html.*

LCD screens hanging on the walls among the original works of the artist. The essential aspects of the idea were the following:

- The computer-generated images should be in the style of Escher's tilings.
- Images should be generated by an evolutionary process, that is, random variation and fitness-based selection on a population of images.
- The visitors should be able to control the evolution by subjective selection. That is, looking at the pictures, they could vote on the images and thereby set the fitness values, defined as the balance between votes in favor of and against a given picture.

Under supervision of H. Labout (Leiden University, Netherlands) and the first author of this chapter, a project team was formed to meet the challenge of making the Escher Evolver.

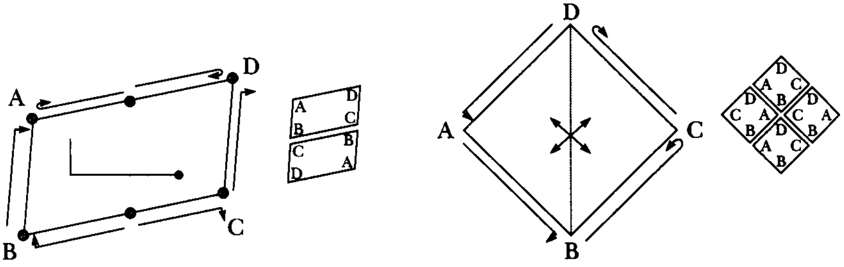
The remainder of this chapter is organized as follows. In Section 17.2 we summarize the mathematical system that can be identified behind the tiling-based works of M. C. Escher. Section 17.3 presents the details of the evolutionary algorithm that has been designed to generate images based on the same system. Section 17.4 discusses the implementation aspects of the software, and the working of the whole system. Here we briefly touch on the behavior of visitors and their reactions to the virtual part of the exhibition. Section 17.5 concludes the chapter by identifying a number of the pitfalls encountered and making recommendations for successful evolutionary art projects.

17.2 THE MATHEMATICAL SYSTEM BEHIND ESCHER'S TILING

A nice overview of the mathematical system behind Escher's tiling is given in Schattschneider (1990, pp. 31–69). Here we briefly recap the most important aspects of it. Escher used two different *ground shapes* when creating his tessellations: triangles or parallelograms. Our evolutionary program uses only parallelograms. Including some special cases of the parallelogram, we have four different ground shapes: the parallelogram, the rhombus, the rectangle, and the square. To fill a plane with one of these ground shapes one can use three different transformations: translation, rotation, and glide reflection (a combination of a reflection and a translation). Using these transformations one can create 10 different transformation systems for filling the plane. These are numbered traditionally as I, II, . . . , X. During this project we did not consider transformation system X since it operates only on triangles. A few translation systems work on all

	I	II	III	IV	V	VI	VII	VIII	IX
A. Parallelogram									
B. Rhombus									
C. Rectangle									
D. Square									

17.2 FIGURE Possible combinations of shapes and transformations.



(a) (b)

17.3 FIGURE Transformation systems (a) II and (b) IV.

ground shapes; the others assume equal side length (rhombus or square) or square corners (rectangle or square). Figure 17.2, after Schattschneider (1990, p. 58), shows all possible combinations of ground shapes and systems. All of these systems are repeating; that is, we can always create a block of these shapes that can fill the plane using only translation.

Figure 17.3(a) shows the details for system II using a simple rotation. Figure 17.3(b) illustrates system IV, which uses two glide reflections. System VI (not illustrated here) is the most complicated, involving two rotations and two glide reflections, used in such a way that a block of 16 shapes is large enough to fill the plane by translation only.

17.3 EVOLUTIONARY ALGORITHM DESIGN

This section describes the evolutionary algorithm (Fogel, Owens, and Walsh 1966; Fogel 2000; Rechenberg 1973; Holland 1973, 1975; Goldberg 1989; Koza

1992; Bäck 1996), which is the computational engine behind the Escher Evolver. It is important to note that in the whole system design it is the representation that forms the greatest challenge. More specifically, the representation determines the syntax (set of genotypes) and its corresponding semantics (set of images encoded by these genotypes), and thus ultimately it determines what kind of pictures can evolve at all. The other components of the evolutionary system (genetic operators, parent selection, population update rule) are more straightforward.

17.3.1 Representation

The evolutionary algorithm operates on a genotype consisting of an array of integers. Several steps are needed to transform these into an Escher-like image. The first step in generating an image is calculating image properties such as size, shape, and colors using the genetic information. These are used to create several affine transformations that transform and paint the building blocks (the tiles forming the basic units of the final image) onto our drawing canvas. Second, the curves forming the contour of a single building block are calculated, and the filling of the block is done. We then have a vector image of a single building block in memory. Using the affine transformations we have created before, they are drawn onto the canvas.

We will now take a detailed look at these steps, beginning with the transformation of the genotype into image properties. Table 17.1 shows the genes we use and the properties they determine.

17.3.2 Ground Shape and Transformation System

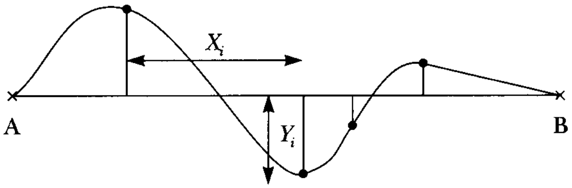
In our genotype we store the ground shape and translation system. Both can be changed independently during crossover and mutation. Therefore we apply a repair algorithm after the reproduction phase. This algorithm reverts an invalid combination to a neighboring valid combination. The vertex displacement genes were to be used for distorting the image by moving the corner points of the shapes, but we have never implemented these displacements. However, Escher has used them in numerous images.

Curves

The contour of a shape is built up from two, three, or four different curves. How many curves are needed, where each curve is placed, and in which direction they

Gene #	Trait	Converted into Range	Note
	Ground Shape		See section on ground shapes
0	Ground shape	Parallelogram, rhombus, rectangle, square	
1	Ground shape angle	amin . . . 180−amin	Parallelogram and rhombus only
2	Edge-ratio	emin . . . emax	Parallelogram and rectangle only
3	Transformation system	1 . . . 10	
4 . . . 5	Vertex displacement		Unused
	Plane Transformation		See section on plane filling
6	Plane scaling factor	smin . . . smax	
7	Plane rotation angle	0 . . . 359	
8 . . . 9	Old color		Unused
	Filling of the Shape		See section on filling the shape
10	Type of view	Top view / side view	
11	Head corner	1 . . . 4	
12	Eye distance	0,1 . . . 0,3	
13	Mouth type	none, loose, beak	Side view only
14	Wing type	none or one of 4 types	
15	Number of wing lines	2 . . . 5	
16	Wing width	0,2 . . . 0,5	
17	Wing length	0,3 . . . 0,8	
18	Tail type	none, full	
19	Number of tail lines	3 . . . 6	
20	Tail edge length	0,2 . . . 0,5	
21	Tail center length	0,2 . . . 0,5	
22	Spine	true, false	Top view only
	Colors		See section on colors
23 . . . 25	Foreground color	HSB–color	
26 . . . 28	Background color	HSB–color	
	Curves		See section on curves
29 . . . 38	Curve 1	3rd-order bezier curve	
39 . . . 48	Curve 2	3rd-order bezier curve	
49 . . . 58	Curve 3	3rd-order bezier curve	
59 . . . 68	Curve 4	3rd-order bezier curve	

17.1 TABLE Representation of the image properties.



17.4 FIGURE Representation of a curve.

are facing are each determined by the transformation system in use. For each of the nine systems we have separately encoded this into the drawing algorithm. To represent the curves we use high-order Bézier curves (see Figure 17.4) because we can use as many control points as necessary (currently we use five of them) and bezier curves have a convex nature, which assures our curves do not cross each other.

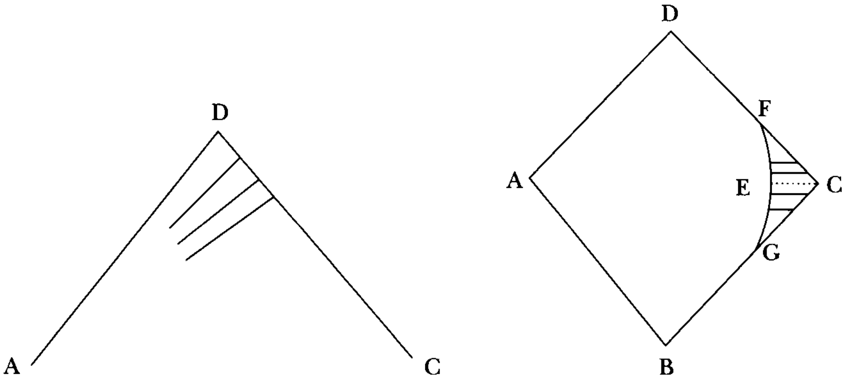
Filling the Shape

Using only a contour, the drawing will never look like an animal such as a bird, fish, or reptile. We have to depict such things as eyes, wings, and tails for this. Our first attempt was to use curves for filling in the shapes, assuming that the subjectively guided evolutionary algorithm would choose curves that tend toward making the figures look lifelike. This didn't work very well, so we settled on a choice of predefined curves to use for filling in the shapes. In the genotype, we have a bit that decides whether the picture is a side view (birds, fish) or top view (fish, reptiles). The top view has two eyes and possibly a spine. The side view has one eye only and can have a mouth. Both types can have several types of wings. Figure 17.5(a) shows a diagonal wing. The direction, number, and length of wing lines are genetically determined. The same applies for the tail section. Here the curve closing the tail, and the number of tail lines are drawn using several genes. Figure 17.5(b) shows a typical tail configuration.

Before drawing, the entire filling is clipped to the shape area, so the wings and tail lines always end at the contour.

Colors

The image has two colors, a foreground and a background color. Both are represented via the HSB (hue-saturation-brightness) color model. Not all values for H, S, and B are possible; for example, the foreground color needs to be a little bit brighter than the background color.



17.5 FIGURE Filling a shape: (a) diagonal wing and (b) tail.

Plane Transformation

After the entire image has been made in vector format, we have to transform it into a bitmap. The plane scaling factor multiplied by a target dependent factor form the size (in pixels) of one shape.

The Result

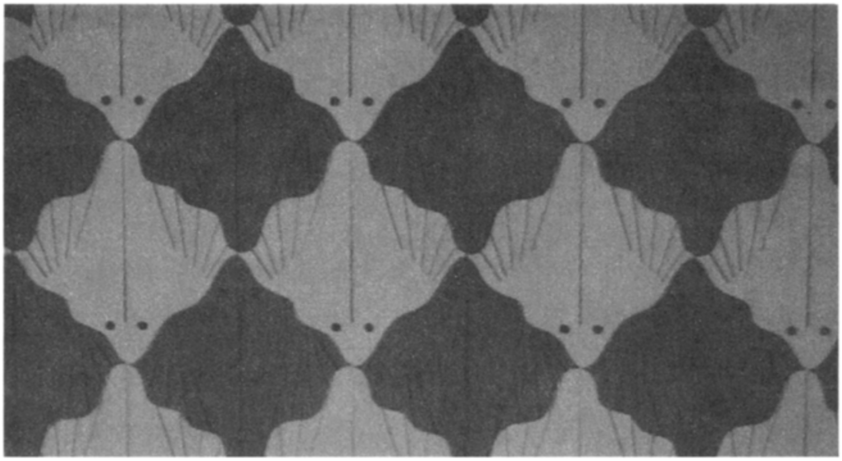
Figure 17.6 shows an evolved tiling that we call "the flatfish."

17.3.3 Genetic Operators: Mutation and Crossover

The genetic operators used are fairly simple. We started with one-point crossover and a mutation operator that may replace a single gene with a random value. Later we developed and used two more sophisticated operators as described next.

N-*Point Crossover*

One-point crossover did not satisfy our needs. For example, using an earlier representation in which the two color genes (for foreground and background colors) were located adjacent to each other, offspring often had both colors (hence same color used for both foreground and background) from a single parent. When uniform crossover was explored, we found that the curves did not cross over very well. We therefore implemented *N*-point crossover (with a random



17.6
FIGURE
The flatfish.

number of crossover points), which provides a better mixture of genes for our purposes, keeping parts of the curve structure intact.

Gaussian Mutation

When we used replacement of genes with random values, strange mutations occurred: Curves arose that were totally different from those contained in either parent, there were radical color shifts, and there were huge size variations. We needed a more subtle approach. Instead of replacing a gene with a random value, a value was chosen from a Gaussian distribution with mean of 0 and a standard deviation equal to 20% of the gene's maximum allowed value. This value is then added to the existing gene value to provide the mutated value.

17.3.4 Selection Mechanism

Experiments during development indicated that tournament selection (as used in the Mondrian Evolver) did not work as expected. However, simple roulette wheel selection performed well, and it was therefore used in this project. During the exhibition we changed from a generational model to a steady-state model (see section 17.4.3), but the selection mechanism remained the same.

17.4 IMPLEMENTATION AND THE WORKING OF THE SYSTEM

There are actually two implementations of the algorithm we have described so far. Both contain the same stand-alone version and a networked version that uses flat LCD screens to gain votes. The main differences between the first and second version are the evolutionary model used (generational versus steady-state) and the way the visitors vote (either for/against a single picture, or via a choice between two pictures). The stand-alone and both networked versions are further described below.

17.4.1 Stand-Alone Version

In the stand-alone version the whole population of six images is shown to one user in one screen. The evolutionary mechanism is completely hidden (that is, the user cannot experiment with different mutation rates, or various crossover types). All the user needs to do is to evaluate the images and press the "next generation" button. This way, the user can quickly see how to direct the evolutionary process toward a certain direction. We also used this version to test the genetic decoder.

17.4.2 First Networked Version

The first networked version, written in April and May 2000, was actually a copy of the stand-alone version. Here, six flat screens have been hung on the walls of the museum, among the regular works, and one image was shown on each screen. Note that here the population size was limited to the number of screens (six), which resulted in a rather small population. Based on this setup a collective of visitors, rather than one user, were evaluating the images, thereby delivering the necessary votes to compute the fitness values of the pictures. Physically, each screen was connected to a client PC, and these clients were all connected to a server.

The work was divided between the server and the clients. The server actually runs the genetic algorithm, and it sends the genetic information to the clients, one individual to each client. The client decodes the genotype it receives and shows it on the screen. The visitors can use two buttons under the screen to vote "yes" or "no" (like this picture or do not like this picture). At a regular interval each client sends its vote count to the server. After an interval of between 5 and

30 minutes (depending on the number of votes received) the server counts all the votes and calculates the fitness of each individual. This is simply the percentage of "yes" votes, using some bias to handle the situation of no votes at all. Finally the server runs the evolutionary algorithm to calculate the next generation, which replaces the current one and is sent to the clients again.

Although the software worked fine, the evolution did not work well. There were not enough votes and there was too much convergence. That is, the population tended to contain pictures that were very similar to each other. The causes for these problems were diagnosed to be in the user interface and algorithm setup:

- The population size of six was too small to hold enough diversity.
- Sometimes—even when running at 30-minute intervals—the total number of votes was only 20 per interval. This resulted in individuals that became very dominant in one or two generations.
- Visitors thought the voting instructions were unclear. Some interpreted them as "Do you like the picture?"; others as "Is the picture Escher-like?". Furthermore, it turned out that people would prefer voting in the context of other pictures, using picture(s) A (B, C, . . .) as a reference to judge picture X. This was impossible because of the distribution of the screens in different rooms in the museum.
- When a visitor voted, he or she received in return only a "thanks for your vote" message as feedback on the screen. This was neither amusing nor interesting enough to motivate them to vote at other screens too. Apparently, people expect an immediate effect after pushing a button.
- The variation in color was not sufficient; we had only light foreground and dark background colors.

We felt that all these factors made the screens not attractive enough. Therefore we started working on a second version that would overcome these problems, while still using the same hardware and genetic representation.

17.4.3 Second Networked Version

Apart from introducing a new color system, which uses three genes per color instead of one, the genetic structure was unchanged. The main novelties were in the user interface and the evolutionary algorithm. We discuss all of the changes below.

Changes to the Evolutionary Algorithm

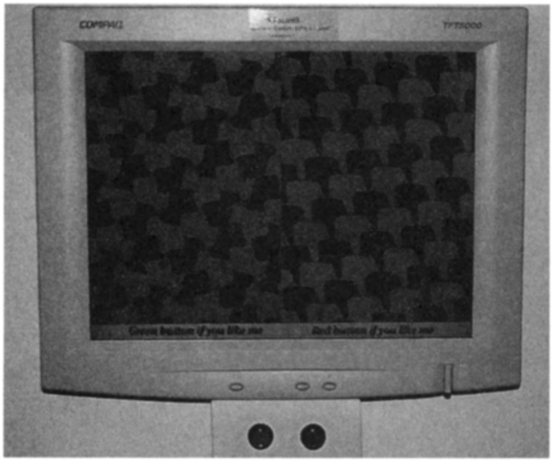
We had to increase both the population size and the number of votes received, but we could only use six flatscreens. To handle the larger population we allocated five individuals to each flatscreen, which made a total of 30 individuals in the population. We could not generate 30 new individuals each generation because we did not have enough votes. A steady-state algorithm—where not all of the population is replaced in one cycle—brought us the solution. In more detail: An individual that stays in the population for many (approximately 5–10) cycles can get many more votes.

In this model the raw fitness of an individual is calculated just as it was in the old model. The fitness values used for reproduction and survival are calculated using age-dependent transformations of the raw fitness values. This protects young individuals from dying quickly and makes old ones reproduce less and die faster. The age correction for reproduction is as follows. The fitness of an individual that has been around in the population for between two and seven generations is transformed to lie between zero and L , where L is an age-dependent limit. L is 100 for individuals two generations old, and reduces by 10 for each age up to seven. There is then a drop to 30 for individuals of age eight, 10 for individuals of age nine, and 0 for any older individuals. Using roulette wheel selection, this means that only individuals between two and nine generations old will ever be selected for reproduction. The correspondence between fitness and survival in the population is treated differently, as follows. Individuals one generation old or younger are given maximum fitness, and so will not be expunged from the population until they are older. An individual two generations old is ensured a survival fitness of at least 60, and individuals three or four generations old have a survival fitness equivalent to their raw fitness. Individuals from five to nine generations old have their survival fitnesses scaled from their raw fitnesses to the interval $(0, Y)$, where Y is 90, 80, 70, 50, and 30 for ages five to nine, respectively, and individuals older than nine generations have a survival fitness of zero.

User Interface Changes

The user interface now has the task of accumulating votes for different individuals. It manages this task by showing two pictures at the same time on a split screen, as seen in Figure 17.7.

The user can vote in favor of one of them. Note that the meaning of the two buttons under the screen changed from "yes/no" for one picture to "yes for left/right" picture. Each time a vote has been made, the client thanks the visitor by replacing the "loser" with an instruction text. A few seconds later two new pictures,



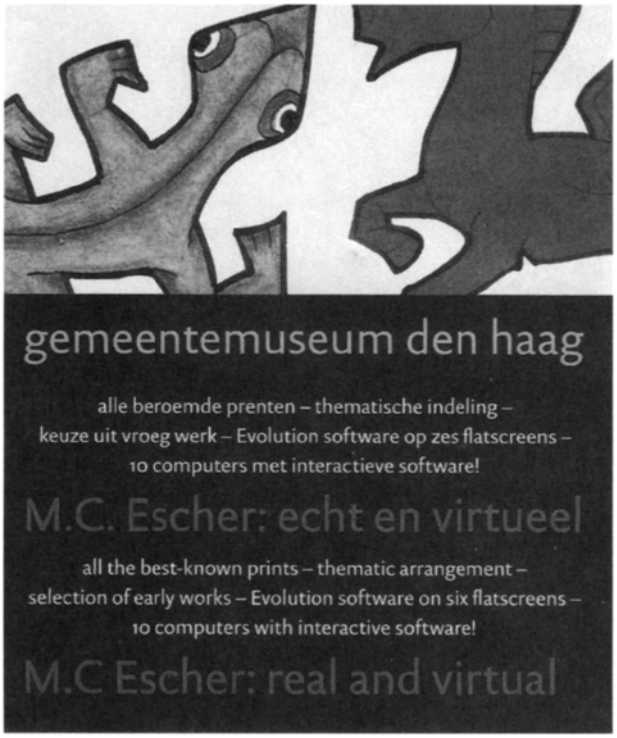
17.7 FIGURE The Escher Evolver at work in the City Museum, The Hague; viewers press the buttons below the display to indicate their preference.

drawn randomly from the pool of five individuals available to the client, are shown. The user may then vote again.

17.5 CONCLUSIONS

The pictures that have emerged during approximately one thousand generations do show some resemblance to Escher's tilings, but they could never be mistaken for a real Escher. The Mondrian showcase was easier in this respect because the underlying structure (straight lines and primary colors) is easier to represent in a computer. We conjecture that there were two factors that prevented the evolution of convincing Escher-like images. One lies in the limitations of the representation, which could be helped by designing more complex genotypes and corresponding decoders. The other is the subjective selection of the ever-changing set of visitors. The collective intelligence, or rather, the collective taste, was arguably not consistent enough to direct the process toward a specific type of image. For the time being, imitating existing art by interactive evolutionary processes remains a great challenge.

The exhibition (Figure 17.8 shows an advertising flyer for it) has been visited by thousands of people, who did actively participate in an evolutionary process of breeding art. In this sense, the project brought (computer) evolution



gemeentemuseum den haag

alle beroemde prenten – thematische indeling –
keuze uit vroeg werk – Evolution software op zes flatscreens –
10 computers met interactieve software!

M.C. Escher: echt en virtueel

all the best-known prints – thematic arrangement –
selection of early works – Evolution software on six flatscreens –
10 computers with interactive software!

M.C Escher: real and virtual

17.8 FIGURE A flyer for the Escher exhibition that included the Escher Evolver.

to the people. The visitors' reactions were mainly positive; criticism was directed to the presented description of the computational mechanisms underlying the virtual part, that is, the Escher Evolver. In fact we were confronted with a trade-off situation. Extensive technical descriptions (tedious and boring) are not read in a museum, but keeping the information limited can lead to an unclear presentation. An interesting aspect was the cognitive experience of the visitors. As discussed in Section 17.4, we noticed a preference for comparing images, as opposed to simply voting for/against one picture. Nevertheless, the overall evaluation of the museum management was very positive; the combination of traditional art and new media, together with the active involvement of the visitors in creating the art shown, have been greatly appreciated.

ACKNOWLEDGMENTS

In addition to those mentioned in this paper, B. Craenen, J. van Hemert, and H. Labout, the authors are grateful to C. Goedhart, N. Huismans, F. den Braber, H. J. Slotboom, U. Wieske, and the Gemeentemuseum staff for their highly valuable contribution to all activities that made this project and this chapter possible.

REFERENCES

Bäck, T. (1996). *Evolutionary Algorithms in Theory and Practice.* Oxford University Press.

Escher, M. C., and F. Bool (1986). *Regular Divisions of the Plane at the Haags Gemeentemuseum.* Haags Gemeentemuseum.

Fogel, D. B. (2000). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* Second edition. IEEE Press.

Fogel, L. J., A. J. Owens, and M. J. Walsh (1966). *Artificial Intelligence through Simulated Evolution.* Wiley.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley.

Holland, J. H. (1973). Genetic Algorithms and the Optimal Allocations of Trials. *SIAM Journal of Computing* 2(2):88–105.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* University of Michigan Press.

Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* MIT Press.

Rechenberg, I. (1973). *Evolutionsstrategie: Optimierung Technisher Systeme nach Prinzipien der Biologischen Evolution.* Fromman-Holzboog Verlag.

Schattschneider, D. (1990). *Visions of Symmetry: Notebooks, Periodic Drawings, and Related Work of M. C. Escher.* W. H. Freeman and Company.

van Hemert, J. I., and A. E. Eiben (1999). Mondriaan Art by Evolution. In E. Postma and M. Gyssens (eds.), *Proceedings of the Eleventh Belgium-Netherlands Confererence on Artificial Intelligence* (BNAIC'99), pp. 291–293.

This Page Intentionally Left Blank

V

PART

EVOLUTIONARY INNOVATION

18 The Genetic Algorithm As a Discovery Engine: Strange Circuits and New Principles
Julian F. Miller, Tatiana Kalganova, Natalia Lipnitskaya, Dominic Job

19 Discovering Novel Fighter Combat Maneuvers: Simulating Test Pilot Creativity
R. E. Smith, B. A. Dike, B. Ravichandran, A. El-Fallah, R. K. Mehra

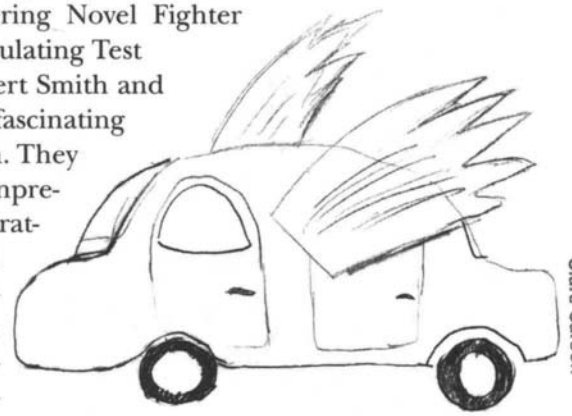
20 Innovative Antenna Design Using Genetic Algorithms
Derek S. Linden

21 Evolutionary Techniques in Physical Robotics
Jordan B. Pollack, Hod Lipson, Sevan Ficici, Pablo Funes, Greg Hornby, Richard A. Watson

22 Patenting Evolved Bactericidal Peptides
Shail Patel, Ian Stott, Manmohan Bhakoo, Peter Elliot

Until recently, many believed that only creative people could invent or originate. Our last part examines some of the newest and most exciting developments in evolutionary computation—the use of evolution to generate totally original solutions to problems automatically.

Julian Miller and his colleagues provide Chapter 18, "The Genetic Algorithm As a Discovery Engine: Strange Circuits and New Principles." Here they explain how evolution can be used to discover original circuit designs instead of simply optimizing existing ones. They suggest that such techniques allow us to find new principles that would be hard to develop through traditional methods.


Clare Carson

Chapter 19, "Discovering Novel Fighter Combat Maneuvers: Simulating Test Pilot Creativity," by Robert Smith and his colleagues, gives a fascinating application for evolution. They show how new and unpredictable aircraft flight strategies can be generated using genetic-based machine learning methods.

The next chapter, "Innovative Antenna Design Using Genetic Algorithms," by Derek Linden, explains another unusual application—the evolution of highly original antennas using a genetic algorithm. This chapter shows the automatic generation of a variety of innovative designs, created with minimal help from the engineer.

Chapter 21, "Evolutionary Techniques in Physical Robotics," by Jordan Pollack and his team at Brandeis University, gives a brief overview of a number of innovative projects, in their goal to achieve fully automated design (FAD) and manufacture high-parts-count autonomous robots (continuing their chapter from the sister book).

The last chapter, "Patenting Evolved Bactericidal Peptides," by Shail Patel and his team at Unilever Research, describes how original proteins can be found (and subsequently patented) by evolutionary search.

18

CHAPTER

The Genetic Algorithm as a Discovery Engine: Strange Circuits and New Principles

Julian F. Miller University of Birmingham
Tatiana Kalganova Brunel University
Dominic Job Napier University
Natalia Lipnitskaya State University of Informatics and Radioelectronics

18.1 INTRODUCTION

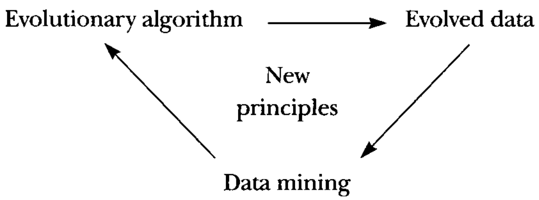
There is a great contrast between the way in which physical systems have been designed by blind evolution and the methods employed by human designers. In the former, entire systems are constructed and tested in situ without a conscious application of principles. In the latter, systems are "evolved" by a process of human ingenuity, which employs a collection of rules, concepts, and principles. It is indeed curious that organisms such as ourselves who are capable of imagining a world that operates according to definite laws and abstract design process were themselves produced by a mechanism that is entirely blind and has no particular object other than survivability. We argue in this chapter that although it is difficult for an evolutionary algorithm to actually suggest new principles directly, new principles may still be inferred by studying an evolved series of examples. We also argue that by employing a blind evolutionary approach, dearly held assumptions and principles may be challenged, and thus, new concepts may emerge.

A well-defined context in which to examine these issues is in the field of electronic circuit design. Here human designers have abstracted the world of binary (an alphabet of *0* and *1*) and multiple-valued quantities (an alphabet of *0, 1*, . . . , *n*) and specified definite operations such as logical OR, AND, and MAX or MIN.

In such a context the *objective* is to construct an electronic or algebraic machine for carrying out a definite function (e.g., addition, multiplication) on a number of input variables, using either as few operations as possible, or a modular construction that can be used to build much larger systems. The usefulness of these electronic machines is readily apparent in modern computers. There is a particular reason why attempting to evolve arithmetic circuits might be a useful and illuminating exercise. Such circuits are *modular* in construction so that very large systems may be constructed from small building blocks. This is clear when we recall that multiplication is a process of repeated addition; thus we can build multiplication circuits by using AND gates to perform elementary one-bit multiplication and then binary full-adders connected in an arrangement called a cellular array. This process is a classic example of human design. First, we construct building blocks that carry out functions that we have abstracted as being fundamental. Second, we build larger systems by manipulating these building blocks by a process of abstract reasoning. When we allow biologically inspired algorithms such as evolutionary algorithms to design the building blocks and assemble the parts, we discover an amazing number of new possibilities. This leads us to the main question studied in this chapter, and it is so important that we shall refer to it as The Fundamental Question (TFQ):

Can we, by evolving a series of subsystems of increasing size, extract the general principle and hence discover new principles?

Such a question immediately leads to many other fundamental questions: What is a subsystem or building block? What is a principle? How can we extract a principle? To be honest, there are not really very precise answers to these questions even in the context of human design. A building block is a subcomponent that has proved useful *by experience.* A principle is really just an observed or deduced rule that is found to be helpful in trying to form a synthesis of a wide range of data and may either directly or indirectly lead to a prediction that can be tested. How are principles discovered? This is the mysterious process of discovery that originates in the human mind and is often called intuition or creativity. It is our firm contention that TFQ will be answered in the affirmative—it is just a matter of time. We hope to show in this chapter that the specialized domain of electronic logic design is a perfect context in which to study this question. We will give a clear example of where we have been able to extract just such a principle (the principle of ripple-carry); however, it is one that humans have known about for a few decades! We should clarify the concept of a principle a little more. Some principles are rules of thumb; others have precise predictive power. Often in science accumulation and study of the former leads inevitably to the latter. Another feature of this chapter is that we show some promising



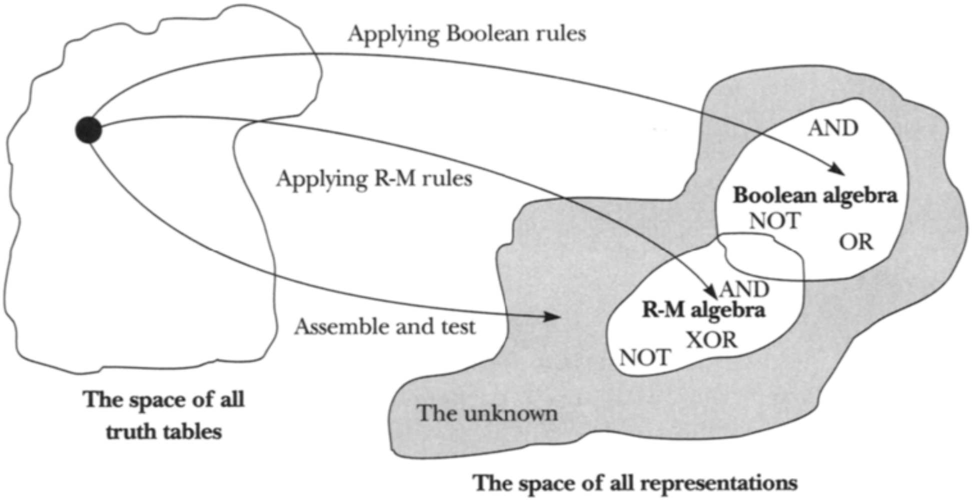
18.1 FIGURE The principle extraction loop.

results that indicate *how* we might be able to extract principles from evolved circuit data. We do this by constructing a "fingerprint" for a class of circuits. Essentially this consists of looking at the relative frequencies of circuit substructures in entire evolved designs. This method and some of our results are discussed in Section 18.5. One can think of this process as one of data mining (DM) evolved systems, and it potentially allows us to "close the loop" of principle extraction, as the extracted principles may feed back into the evolutionary algorithm (Figure 18.1).

18.2 THE SPACE OF ALL REPRESENTATIONS

Every binary and multiple-valued function is specified by a truth table. The truth table specifies what values the outputs of a function are for all values taken by the function inputs. There are certain special collections of operators that act on a binary or multiple-valued function that have the property that *any* function can be represented by expressions involving these operators and the input variables. The collection of these operators and the sets they operate on is often referred to as an *algebra*. In the case of binary functions, there are two well-known algebras: Boolean, which uses AND, OR, and NOT, and Reed-Muller (R-M), which uses AND, XOR and NOT. Multiple-valued logic also has its own algebras, and often they are referred to as *functionally complete bases*. When these algebras are used, a given function can only be represented by a particular class of expressions. The basic concept is shown in Figure 18.2.

The unknown region in Figure 18.2 depicts all the representations of logic functions written as an expression that does not use operations taken from the set {NOT, AND, OR, XOR}. Any expression in this region *once known* could be manipulated to become either an expression in the R-M or Boolean regions. To clarify the concept of a representation we give a simple example. Suppose we


Applying Boolean rules
AND
Boolean algebra
NOT
OR
Applying R-M rules
AND
R-M algebra
XOR
NOT
Assemble and test
The space of all truth tables
The unknown
The space of all representations

18.2 FIGURE How assemble-and-test reaches the unknown regions of the space of all representations.

have a truth table that we discover by assemble-and-test can be represented quite simply as the output y given below:

$$y = \overline{f}x_4$$

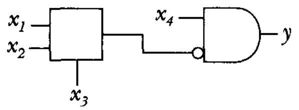
where f is the output of a MUX gate with inputs x_1 and x_2 and control input x_3. This is represented in circuit form in Figure 18.3.

If this function were to be represented in the Boolean region of Figure 18.2, it would have the expression given by the following equation, where we have represented the NOT operation by an over bar, the OR operation by +, and the AND operation is assumed between literals x_i.

$$y = \overline{x}_1\,\overline{x}_2 + \overline{x}_1\,\overline{x}_3 + \overline{x}_2\,\overline{x}_3 + \overline{x}_4$$

Implementing this expression as an actual circuit would require six logic gates (three ANDs and three ORs), not including the NOT gates required.

It would be extremely difficult to develop an algorithm that did not use the assemble-and-test concept but would be able to synthesize circuits of the form shown in Figure 18.3. One of the fundamental contentions of this chapter is that



18.3 FIGURE Example circuit corresponding to equation $y = \bar{f}x_4$.

the assemble-and-test method is the *only* way that the space of all representations can be explored.

Although we have discussed the design of logic circuits, the idea expressed in Figure 18.2 could be regarded as an analogy for the design process in general. The space of all representations would then become the space of all designs.

18.3 EVOLUTIONARY ALGORITHMS THAT ASSEMBLE ELECTRONIC CIRCUITS FROM A COLLECTION OF AVAILABLE COMPONENTS

The idea of building electronic circuits by using an evolutionary algorithm to connect logic gates together from a set of possible types has only recently begun to emerge in recent years. Using the assemble-and-test methodology to try to get whole circuits to perform specific tasks rather than employing complex human design principles is one of the important themes that has arisen in the nascent field of "evolvable hardware" (Sipper et al. 1997). Many different approaches to this have been developed. Iba, Iwata, and Higuchi (1996) showed how boolean combinational logic functions (not involving time) could be built using a genetic algorithm to evolve the connections between AND, OR, and NOT gates. Thompson (1997) showed how circuits could be synthesized on special devices called field-programmable gate arrays (FPGAs) to carry out frequency discrimination tasks and robot control. The basic technique was to evolve binary configuration sequences to program an FPGA to carry out the desired task. There was no human input about *how* this might be done, just a measurement of the degree to which a given circuit achieved the desired response. Remarkably it was shown that tiny circuits could be evolved to carry out the task efficiently, but that operated in ways that are still not understood, and still under intense investigation. It was clear that the evolved circuits were making use of all properties that were potentially useful, including the physical properties of the silicon medium. In our own work in this field (Miller, Thomson, and Fogarty 1997; Miller and Thomson 1998a, 1998b, 1998c) we have designed a powerful technique for evolving the choice of functions and the connections of an array of digital logic functions. The general concept of this can be envisaged with a simple example.

In Figure 18.4 a gate array representation of an evolved one-bit adder is given. The inputs A, B, and Cin are the binary inputs. The outputs Sum and Cout are the binary outputs. Sum represents the sum bit of the addition of A + B + Cin, and Cout the carry bit. The chromosome representation of this is

0 1 0 **10** 0 0 2 **6** 3 2 1 **10** 0 2 3 **16** 6 5

The figures in bold represent the functions of the corresponding logic cells. The allowed cell functions can be chosen to be any subset of those shown in Figure 18.5, where ab implies a AND b, $\bar{a}$ indicates NOT a, $\oplus$ represents the exclusive-OR (XOR) operation, and + the OR operation.

Functions 0–15 are the basic binary functions of 0, 1, and two inputs. Functions 16–19 are all binary multiplexers with various inputs inverted. The multiplexer (MUX) implements a simple IF-THEN statement (i.e., IF c = 0 THEN a ELSE b).

An evolutionary algorithm is used to evolve circuits by beginning with a population of randomly initiated chromosomes. In some cases this was a genetic algorithm (GA) with uniform crossover (50% genetic exchange). The chromosomes are constrained so that columns of cells can only connect to cells to their left. This is necessary to ensure that the resulting circuits are not time dependent (feed-forward, combinational). Sometimes the evolutionary algorithm employed was a simple form of $(1 + \lambda)$ evolutionary strategy (ES). In this case a population of $(1 + \lambda)$ random chromosomes were randomly generated and the fittest chromosome selected. The new population is then filled with mutated versions of this. Random mutation was defined as a percentage of genes in the population that were mutated. The mutation operator respected the feed-forward nature of the circuits and also the different alphabets associated with connections and functions.

18.3.1 Binary Circuit Symbols

Figure 18.6 shows the symbols used to represent gates in binary circuits. Note that on some diagrams a small circle may be seen on an input or output wire; this indicates that the input or output is inverted (by applying a NOT operation). This notation is also used in the multiple-valued case.

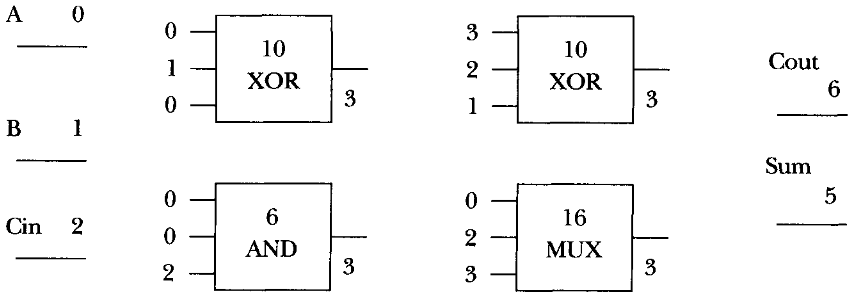


FIGURE 18.4 Gate array representation of evolved one-bit adder.

0	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**
0	1	a	b	$\bar{a}$	$\bar{b}$	ab	$a\bar{b}$	$\bar{a}b$	$\bar{a}\bar{b}$

10	**11**	**12**	**13**	**14**
$a \oplus b$	$a \oplus \bar{b}$	$a + b$	$a + \bar{b}$	$\bar{a} + b$
15	**16**	**17**	**18**	**19**
$\bar{a} + \bar{b}$	$a\bar{c} + bc$	$a\bar{c} + \bar{b}c$	$\bar{a}\bar{c} + bc$	$\bar{a}\bar{c} + \bar{b}c$

FIGURE 18.5 Allowed cell functions.

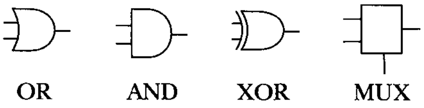
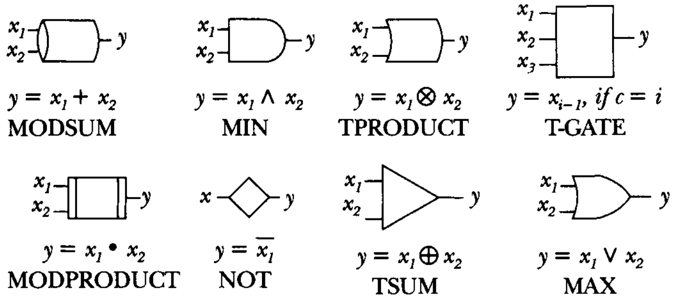


FIGURE 18.6 Binary circuit symbols.

18.3.2 Multiple-Valued Circuits

The methods used in the section above have also been extended to multiple-valued circuits. Multiple-valued logic is an extension of the more familiar Boolean logic to an alphabet of positive integers 0, 1, . . . , *n*, where *n* is referred to as the *radix*. In this logic the familiar operators such as logical a AND b, a OR b, are replaced by the minimum of {a, b} and maximum, respectively. Many other operators can be defined that have no counterpart in binary. The details of this



18.7 FIGURE Symbols and analytic representations of two-input multiple-valued logic gates.

approach have been reported in (Kalganova, Miller, and Fogarty 1998; Kalganova, Miller, and Lipnitskaya 1998). The following logic gates have been used in circuit evolution: NOT, MIN, MAX, MODSUM, TSUM, TPRODUCT, and any of these with the output inverted (excluding NOT).

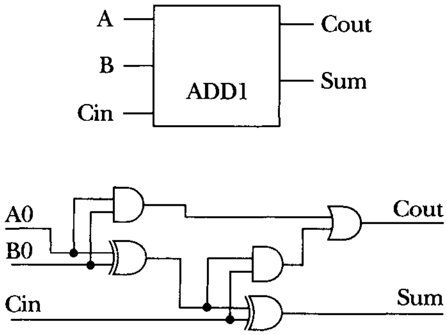
In Figure 18.7 various diagrammatic representations of multivalued functions are given together with their symbolic expressions. Implementations of these gates can be found in Jain, Bolton, and Abd-El-Barr (1993). We also use the three-valued T-gate as one of the basic components for the design (Kameyama and Higuchi 1986). The T-gate is the multiple-valued counterpart of the binary multiplexer.

18.4 RESULTS

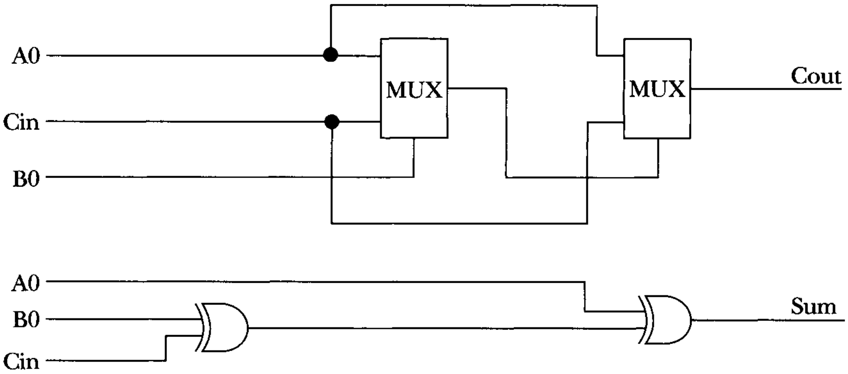
Having examined the evolutionary algorithms and related topics, we now focus on some circuits evolved using these methods. The following sections describe five such circuits: one-bit adders, two-bit adders, two-bit multipliers, three-bit multipliers, and, finally, multiple-valued one-digit adders with carry.

18.4.1 One-Bit Adder

The conventional one-bit adder is depicted in Figure 18.8. The conventional (most efficient) circuit diagram requires five logic gates of three types (AND, OR, XOR). Note that the sum output is implemented with XOR gates. This is a characteristic of the human design.



18.8 FIGURE Block diagram of one-bit adder with carry and conventional circuit diagram.



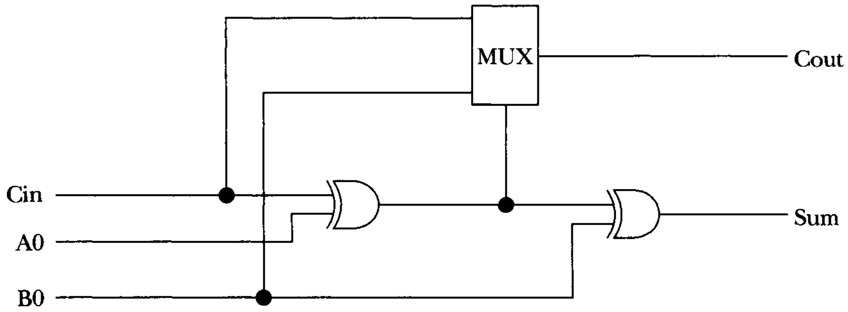
18.9 FIGURE Evolved one-bit adder with carry (A).

When an evolutionary algorithm is used, some strange but quite elegant solutions are possible. Figure 18.9 depicts an efficient one-bit adder that was obtained. In reality, MUX gates can be built out of smaller components, two ANDs and one OR. However, some modern devices use the MUX gate as an "atomic" device in that all other gates are synthesized using this. The MUX gate is unusual in this respect in that by merely setting one of the three inputs to 0 or 1 it can realize any binary function of two variables.[1]

An interesting feature of this circuit is that the sum and carry parts of the circuit are completely decoupled and there is a nice symmetry to it, which suggests

---

[1] Actually there are precisely six functions that have this special property; they are often called universal logic modules (ULMs).



18.10 FIGURE Evolved one-bit adder with carry (B).

that just as XOR gates naturally carry out elementary addition, so the MUX gate naturally synthesizes the carry process of addition. This might be thought of as a potential new principle that has not been observed by human designers. It should be noted that the circuit was actually evolved by separating the carry function from the sum and evolving them separately. When both are evolved together under a more constrained geometry (2 × 2 cells), it becomes more difficult for the EA to correctly synthesize the circuit, and it requires about 20 runs of 2,000 generations (population size 50, 100% breeding, 5% mutation) to produce a solution (Figure 18.10).

This is a remarkably efficient circuit (its genome was explained in Section 18.3). Actually this circuit *is* known but was only recently discovered.[2]

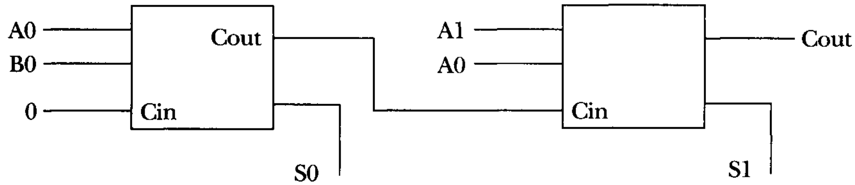
18.4.2 Two-Bit Adder

One of the principles used by humans to construct larger adders is known as the *ripple-carry* principle. The block diagram for a two-bit adder is shown in Figure 18.11. Each of the blocks in the figure is identical to that depicted in Figure 18.8. The two-bit adder can perform the calculation $a + b$, where a and b are any positive integers between 0 and 3. We were able to evolve the two-bit adder without the insertion of any hints about how to do it. In the first design shown in Figure 18.12, we again separated the sum part of the circuit from the carry part.

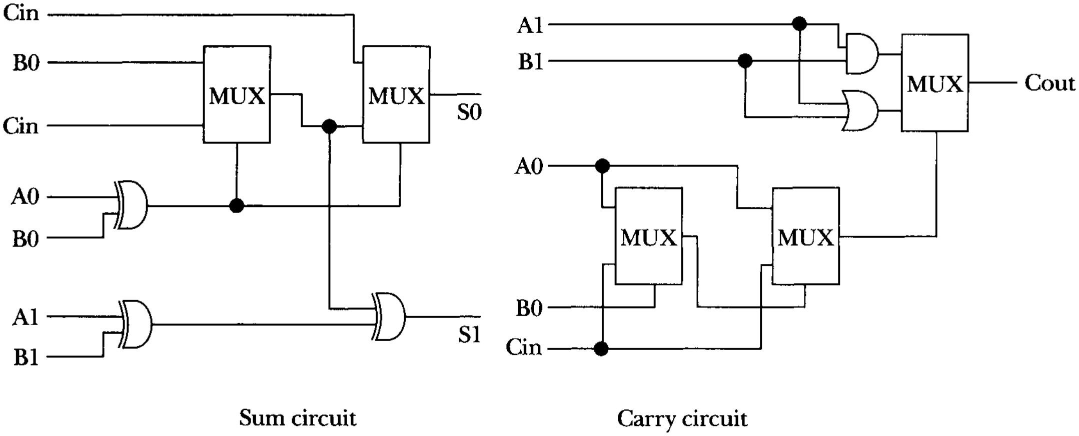
We did this so that we could see if the new evolved circuit has anything in common with the evolved one-bit adder (Figure 18.9). It is clear that they do as the carry circuit of the one-bit adder is actually used as an important part of the

---

[2] Personal communication by John Gray, formerly of Xilinx, Edinburgh.


A0
B0
0
Cout
Cin
S0
A1
A0
Cin
Cout
S1

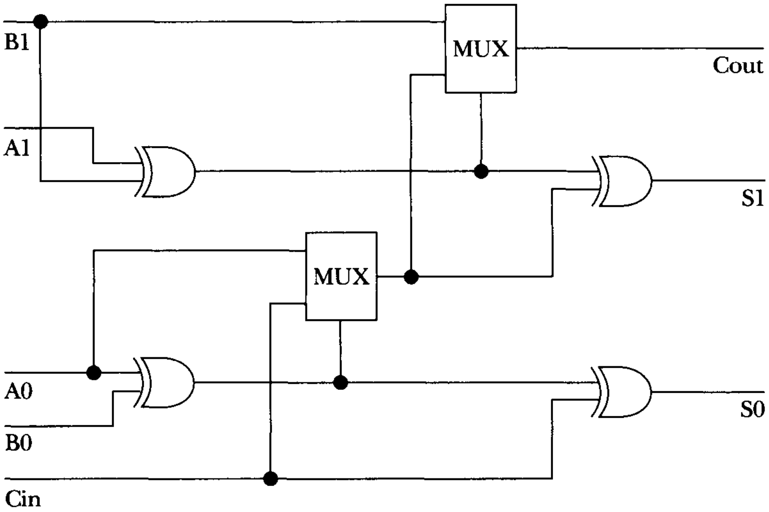
18.11 FIGURE The two-bit ripple-carry adder.


Cin
B0
Cin
A0
B0
A1
B1
MUX
MUX
S0
S1
A1
B1
A0
B0
Cin
MUX
MUX
MUX
Cout
Sum circuit
Carry circuit

18.12 FIGURE Evolved two-bit adder (A).

carry circuit for the two-bit adder. There is a lot of subtlety to this! As before with the one-bit adder we subsequently evolved the complete adder without the carry being separated. Again the task was more difficult. A 3 × 3 geometry was chosen, and the genetic algorithm had the following parameters: population size 50, 50,000 generations, 20 runs, breeding 100%, mutation 5%, elitism, levels-back = 2. In the 20 runs, five solutions were obtained that were 100% functional. The number of required cells were 6 (1), 7 (1), 8 (3). The best circuit is shown in Figure 18.13.

Comparing this circuit with the best evolved one-bit adder circuit (Figure 18.10) it can be seen that the two-bit adder circuit is produced by connecting the two smaller adders in a configuration identical to that shown in Figure 18.11. Clearly we can then deduce the modular design principle of the ripple-carry adder and hence build adder circuits of *any* size. This demonstrates that principles



18.13 FIGURE Evolved two-bit adder (B).

		A2	**A1**
		B2	B1
		A2B1	A1B1
	A2B2	A1B2	
P4	P3	P2	P1

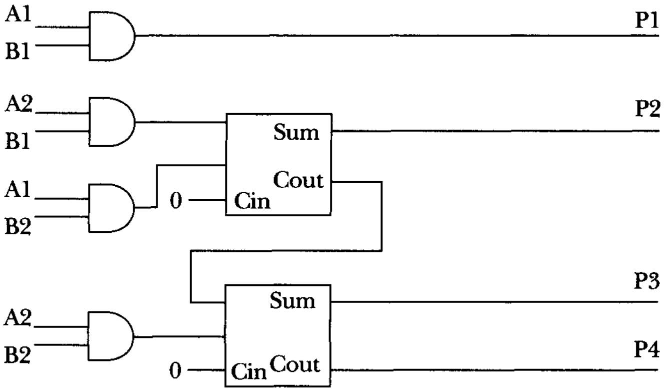
18.14 FIGURE Multiplication of two-bit binary numbers.

can be extracted from a series of evolved examples. However, in this case human designers already know the principle!

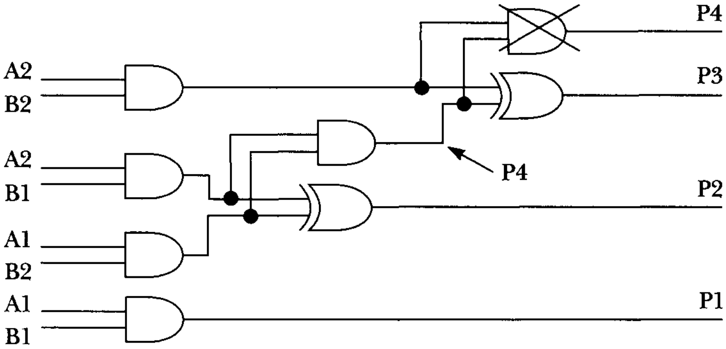
18.4.3 Two-Bit Multiplier

The process of designing a circuit to perform multiplication is modeled on the familiar long multiplication process shown in Figure 18.14.

The multiplication of two one-bit binary numbers is accomplished with an AND gate. Thus to build a circuit to multiply two integers a, b (between 0 and 3), one requires the circuit shown in Figure 18.15.



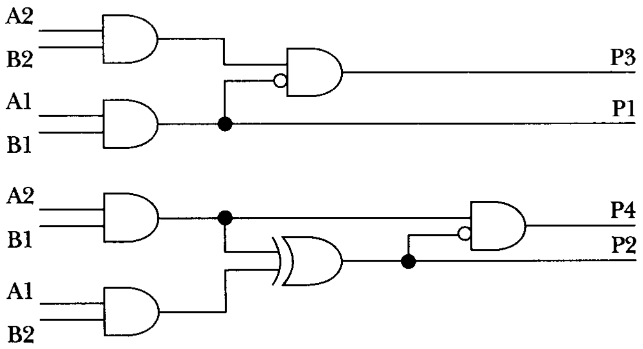
18.15 FIGURE Two-bit cellular multiplier.



18.16 FIGURE Gate diagram for most efficient two-bit multiplier.

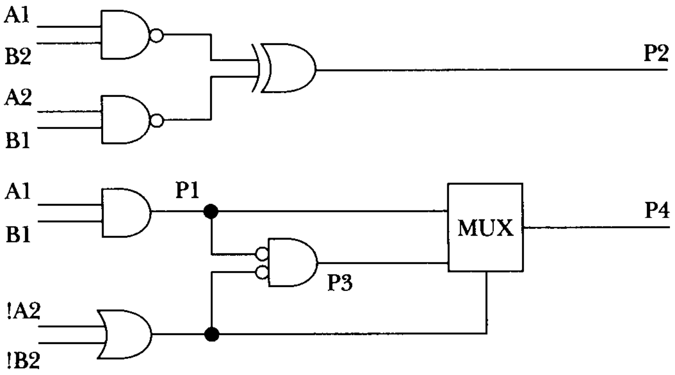
A gate-level picture of this is given in Figure 18.16. The X on the topmost AND gate indicates that in practice this gate is not required, as the most significant bit of the output product (P4) can be obtained without it.

To obtain a plentiful supply of fully functional evolved two-bit multiplier circuits a (1+3)-ES algorithm was used with uniform mutation. The mutation probability was 0.02. One thousand runs of 50,000 generations were carried out with a 3 × 4 geometry and allowed gates types 6–16 (see Figure 18.5). The levels-back parameter was set to 4. Of the 1,000 runs, 992 circuits were produced that were 100% correct. Of these, 139 required only seven gates and so were as efficient as the conventional circuit shown in Figure 18.16. Three of the evolved circuits are shown in Figures 18.17 to 18.19.


A2
B2
A1
B1
A2
B1
A1
B2
P3
P1
P4
P2

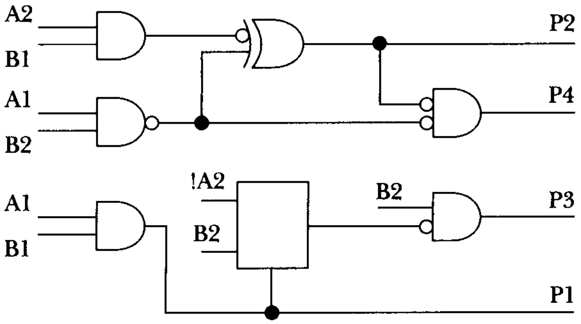
18.17
FIGURE

An evolved two-bit multiplier (A).


A1
B2
A2
B1
P2
A1
B1
P1
MUX
P4
P3
!A2
!B2

18.18
FIGURE

An evolved two-bit multiplier (B).


A2
B1
A1
B2
P2
P4
A1
B1
!A2
B2
B2
P3
P1

18.19
FIGURE

An evolved two-bit multiplier (C).

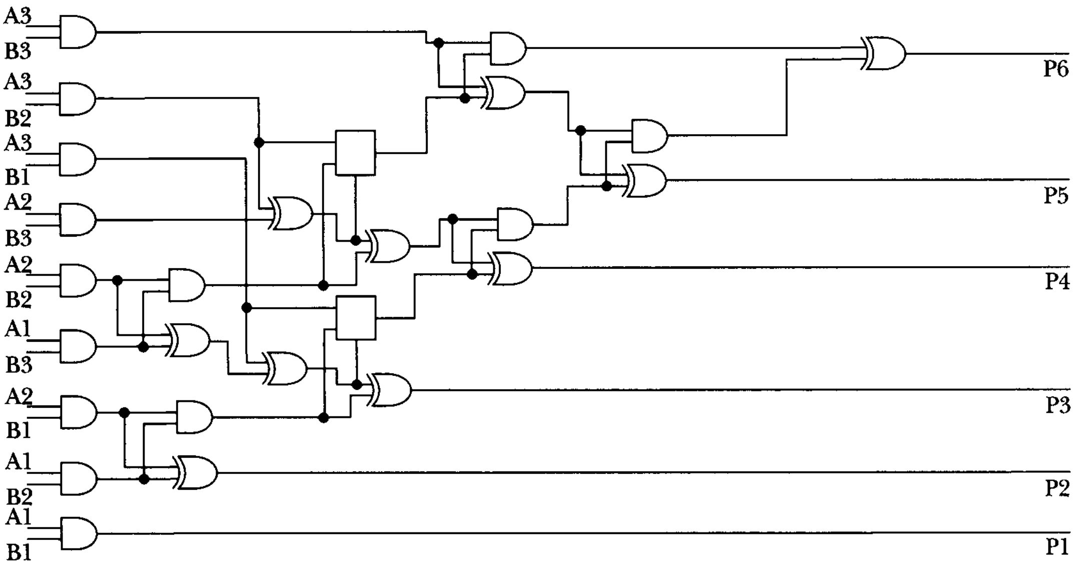
Examining these circuits instantly reveals their strangeness. In circuit A there are two independent subcircuits, one involves P1 and P3, and the other, P2 and P4. This is very counterintuitive as in the conventional model of multiplication (Figure 18.16) none of the outputs are reused (except possibly P4, which turns out to be an input to the XOR gate with P3 as the output). P3 is produced by three gates in circuit A, whereas it needs four in the conventional circuit. The circuit for P2 in all three evolved circuits is effectively the same. In the conventional circuit P1 has nothing to with P3, yet in two of these circuits (A and C) P1 is used to produce P3. Despite the fact that one can apply the symbolic rules of Boolean algebra to show that all these circuits are transformations of one another (including the conventional), the calculations are not obvious at all. Another strange feature of the circuits—particularly circuit A—is that only *one* XOR gate is required yet there are *two* additions. It would appear that our human concept of addition as only being modeled by the XOR operation is not correct.

Since our desire here is to answer TFQ in the affirmative, we looked at the problem of evolving the three-bit multiplier. Could we discern any principles at work in the evolved two-bit multipliers that also operated in the evolved three-bit multipliers, which would enable us to build efficient multipliers of any size?

18.4.4 Three-Bit Multiplier

When the conventional three-bit cellular multiplier is represented at gate level and all redundant gates are removed, the circuit shown in Figure 18.20 is obtained. This circuit can calculate the product of two integers a and b in the range 0–7. In our experiments we chose to use just the gates ab, a$\underline{b}$, and a^b (see Figure 18.5) because we felt that the evolved two-bit multiplier shown in Figure 18.17 was quite elegant and involved only three gates and also because of the "fingerprinting" methods explained in Section 18.5. We felt that if we examined evolved three-bit multipliers with just these gates we might stand a better chance of deducing some general principles. To obtain a reasonable probability of obtaining 100% correct solutions it was found that one had to evolve around 3,000,000 generations. We used a geometry of 6 rows and 7 columns with the maximum possible levels-back (7). The mutation probability was chosen to be 0.02.

We ran the ES 550 times and obtained 178 100% functional circuits. Figure 18.21 shows the number of circuits that required 30 gates or fewer. There were 58 circuits of this type. The most efficient conventional circuit shown in Figure 18.20 requires 30 two-input gates. Thus all 58 circuits are at least as efficient, and 29 are more efficient. Note that in this experiment we did not use multiplexers. It is therefore possible that we could obtain even more efficient circuits.

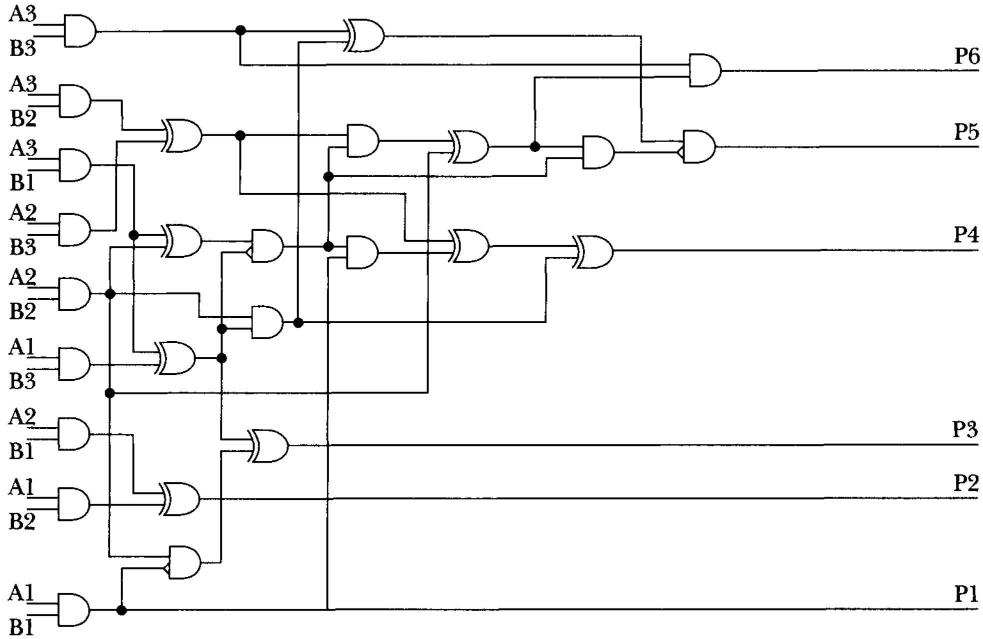


18.20 FIGURE Most efficient conventional gate-level three-bit multiplier (30 two-input gates, 26 with MUXs).

Gates/circuit	30	29	28	27
Number of circuits	29	17	11	1

18.21 FIGURE Number of gates used for 100% functional circuits.

The most efficient evolved three-bit multiplier, shown in Figure 18.22, requires only 26 gates rather than 27. This is because when we examined the evolved circuit we discovered that one of the gates used was logically redundant and could be removed. When we compare the evolved three-bit multiplier with the conventional, we notice that there are considerable differences and it is not possible to see whether there are repeating modules when looking at the evolved circuit. However, we can see that outputs P_1 and P_2 are implemented in the same way in both circuits. In the evolved circuit P_1 is used twice, whereas in the conventional circuit it is never used again. P_2 is not reused in either circuit. P_3 requires seven gates in the evolved circuit in comparison with nine gates for the conventional circuit. One of the interesting features of this design is that in the evolved circuit P_3 depends on P_1. This does not happen in the conventional circuit. The same is true of output P_4; it too depends on P_1 in the evolved circuit.



18.22 FIGURE Evolved three-bit multiplier (most efficient—26 gates).

The differences between the two circuits were very marked when we looked how the outputs P_k depended on the elementary products (A_iB_j). First, we noticed that in conventional design A_1B_1 is not used by any other outputs. However, in the evolved circuit this elementary product is used in the implementation of P_3 and P_4. The elementary products (A_2B_1) and (A_1B_2), which are used in the implementation of P_2, are not involved in any other outputs in the evolved circuit. However, in the conventional implementation these products are used for every output except P_1.

It is clear that the way the multiplication process is modeled in the evolved circuit is very different from the human one.

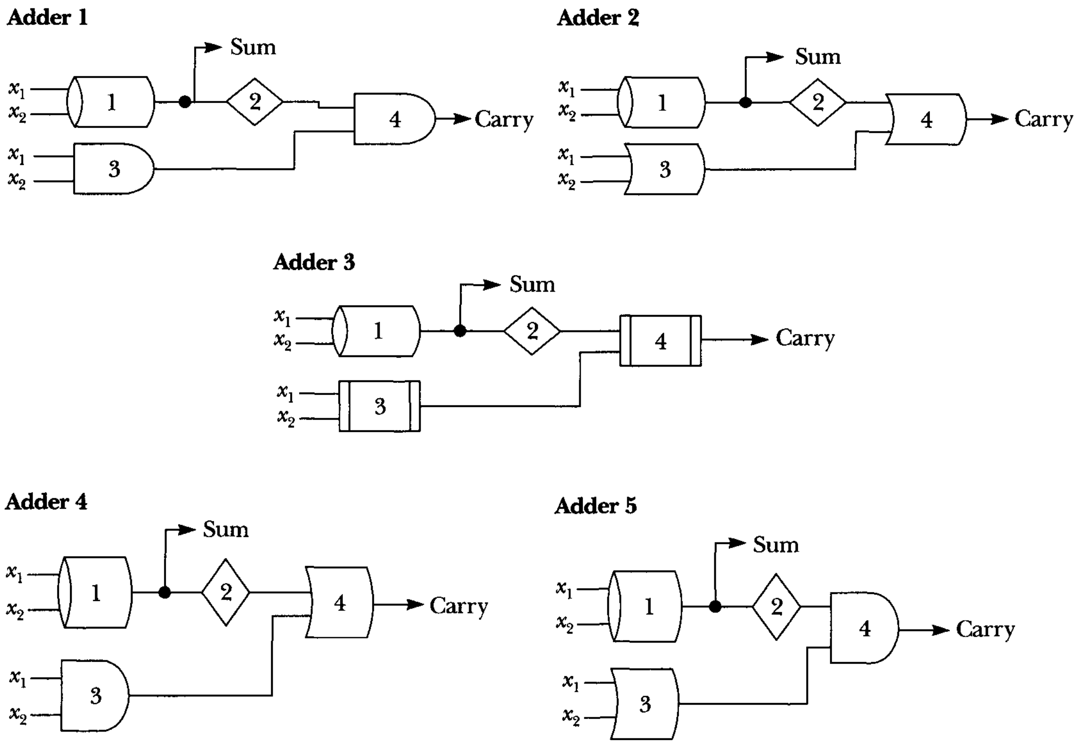
18.4.5 Multiple-Valued One-Digit Adder with Carry

Multiple-valued logic contains a lot of different algebras that we can use to represent a given multiple-valued logic function. Each of these logic algebras contains a specific set of multiple-valued logic operators and is called in the literature a *functionally complete basis*. This means that it can implement any multiple-valued logic function of *n* variables. As we saw in Section 18.2, there is a space of expressions that represents a given function. This function can be represented by any of the functionally complete bases. There are well-known human-developed tools to map this function using the specific functionally complete basis into a given expression. However, we encounter problems when we want to combine some of these functionally complete bases to represent a function in an economical way. Traditionally this would mean that we would have to develop a specific logic algebra; this takes a lot of time and effort, and there is no guarantee of success at the end of the process. In addition it is only after this procedure that we are able to map the logic function into a given basis. This is one of the disadvantages of the human design procedure. In order to overcome this difficulty we, as in the binary case, adopted the method of assemble and test. This allows us to use *any* of the logic sets and in principle obtain *any* possible logic expression. Note that the set used should contain one of the functionally complete bases known in order to guarantee success. Thus we potentially can discover new and highly efficient alternative representations that are counter to human intuition.

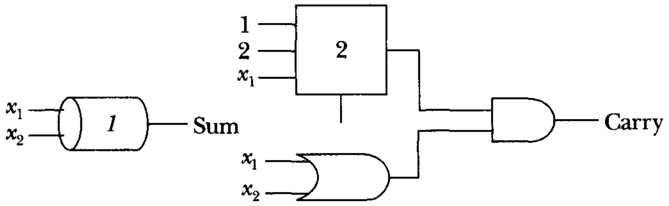
Here we will look at some evolved designs for the one-digit three-valued adder with output carry. Three-valued logic functions can take the values in the set {0, 1, 2}. Thus, for example, in terms of base 3 arithmetic, $1 + 2 = 10$; that is, the sum is 3 carry 0. In Figures 18.23 and 18.24 we give some examples of evolved circuits in the nonstandard set of multiple-valued gates. Note that within any of the sets discussed we always used a functionally complete basis as a subset of the gates chosen.

Figure 18.23 shows some evolved designs for the one-digit three-valued adder with carry. All these circuits have the same structure. The difference between them lies with gates 3 and 4. Analysis of the resulting logical expressions gives rise to equations that are extremely difficult to prove in a purely algebraic manner. However, because the assemble-and-test method doesn't explicitly carry out formal algebraic operations, we find such unusual structures relatively easily.

The circuit shown in Figure 18.24 involves four different types of gates. We were unable to find any multiple-valued logic design methods that allowed us to represent the function in this way. The algebra of multiple-valued logic is still incomplete, but it is clear that using the assemble-and-test method allows us to escape from the restrictions inherent in a particular algebra.


Adder 1
Sum
x_1
x_2
1
2
3
4
Carry
Adder 2
Sum
x_1
x_2
1
2
3
4
Carry
Adder 3
Sum
x_1
x_2
1
2
3
4
Carry
Adder 4
Sum
x_1
x_2
1
2
3
4
Carry
Adder 5
Sum
x_1
x_2
1
2
3
4
Carry

18.23 FIGURE Evolved one-bit adder circuits that are identical except for gates 3 and 4.


1
2
x_1
2
x_1
x_2
1
Sum
x_1
x_2
Carry

18.24 FIGURE Evolved one-bit adder with T-gate.

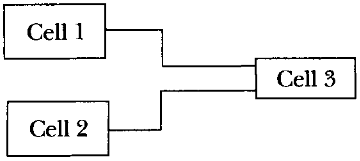
18.5 FINGERPRINTING AND PRINCIPLE EXTRACTION

There are many types of principles that potentially might be extracted from a database of genotypes of evolved circuits. In Figure 18.1 we saw that after the stage of collecting evolved data, we are faced with a problem of data mining. The extraction of knowledge from a large collection of evolved data is a little like trying to identify which genes (in terms of base sequences of DNA) are responsible for particular inherited characteristics. It is not an easy task. One approach to this is to try to categorize the various types of subcircuit that are present in the evolved genotypes. Since there are many subcircuits that are permutations of one another, we must find a way of normalizing the data so that the permutations are readily identifiable. Additionally we must look for subcircuits of a particular form and size. This is necessary to avoid the combinatorial explosion that would occur if we wished to enumerate all possible subcircuits. As a first attempt in this direction we decided to analyze the evolved genotypes (after permutational normalization) in terms of 2-into-1 subcircuits.

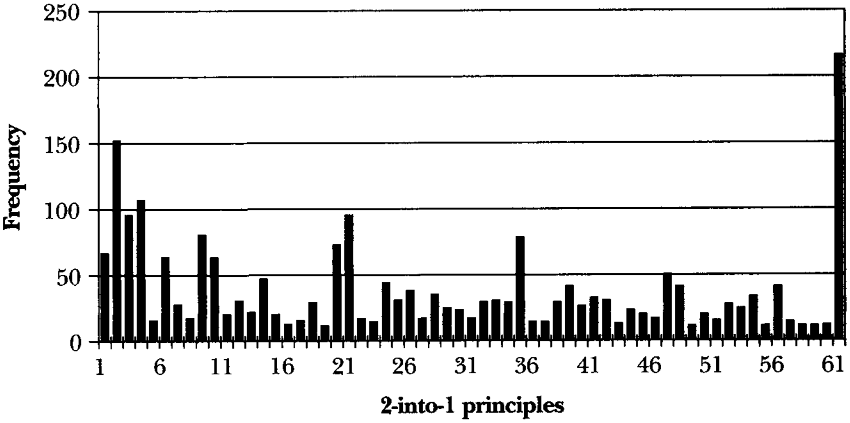
The concept of a 2-into-1 subcircuit is shown in Figure 18.25. In the figure each cell may be any logic gate including a multiplexer. Each multiplexer would therefore have six 2-into-1 subcircuits associated with it (as it has three inputs). Also if cell 3 was an $a\underline{b}$ gate (type 7) there would be two different subcircuits of form XY7 and YX7 for each pair of X Y values (X,Y are allowed gate types in the genotypes). To avoid this explosion of possible subcircuits, we ignored the particular inputs that cells 1, 2, and 3 were connected to. We then indexed all possible XYZ triples and collated the frequencies of occurrence of these in the evolved material. We analyzed the evolved genotypes associated with the 100% solutions for the two-bit multiplier. There were 61 distinct 2-into-1 principles that occurred more than 10 times. Figure 18.26 shows a histogram of these.

The seven highest peaks correspond to the subcircuits shown in Table 18.1. The 6-6-10 subcircuit corresponds to two ANDs into XOR. In human design terms one would see this as adding two multiplications, so we would not be very surprised to see this being an important principle in the multiplication process. The 6-6-6 subcircuit could be seen as connected with the "carrying forward" operation. The third most frequently occurring is 6-15-7.

Actually gate 15 is logically identical to a NAND gate, thus 6-15-7 is a close relative of 6-6-7. Also 6-6-8 is a close relative because gate 8 is the same as gate 7 with the inputs reversed. Clearly the 6-6-7 subcircuit is very important. Yet this structure is not involved in the conventional multiplier at all (Figure 18.16). It can be seen in the evolved example shown in Figure 18.17. One can think of the histogram shown in Figure 18.26 as being a circuit "fingerprint." Admittedly it is



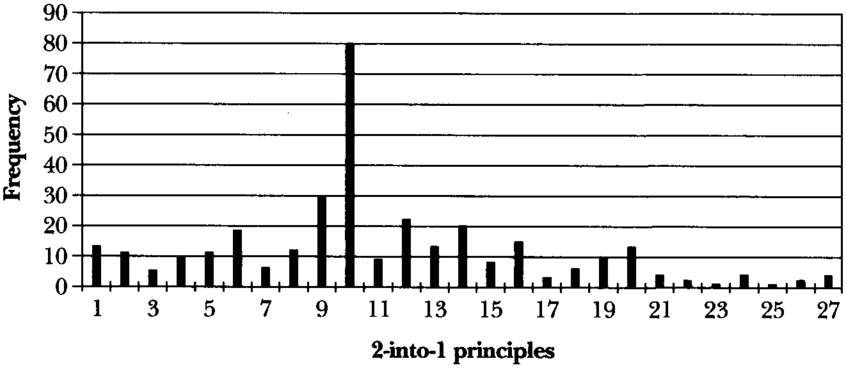
18.25 FIGURE The 2-into-1 subcircuit principle.



18.26 FIGURE Distribution of 2-into-1 principles occurring in evolved two-bit multiplier genotypes (with frequency greater than 10).

Cell 1	Cell 2	Cell 3	Frequency
6	6	10	215
6	6	6	151
6	15	7	107
6	6	7	94
6	6	8	94
15	6	8	80

18.1 TABLE Seven most frequently occurring subcircuits.



18.27 FIGURE Distribution of 2-into-1 principles occurring in evolved three-bit multiplier genotypes.

just one of a number of possible subcircuit histogram plots, but it is probably one of the most fundamental.

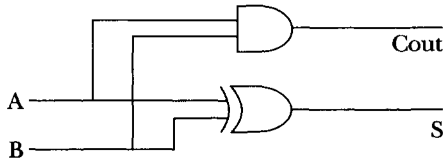
Inspired by the obvious usefulness of gates 6, 7, and 10 (AND, AND with inverted input, and XOR) we evolved many 100% solutions for the three-bit multiplier where we had allowed only these three gate types. The evolved three-bit multiplier shown in Figure 18.22 is an example of this. It is clear one should distinguish between inputs to obtain a better picture. This was carried out for the analysis of three-bit multiplier solutions that were 100% correct using gates 6, 7, and 10 only. The fingerprint is shown in Figure 18.27.

The highest peak corresponds to 6-6-10, with 80 occurrences. The next highest peak corresponds to 6-6-7, with 30 occurrences.

There is still much further work to be done here. A better way of trying to characterize 2-into-1 principles would be to look at logical behaviors as this would remove the occurrence of subcircuits that have different gene triples but the *same* behavior. Also one needs to take into account that useful modular subblocks may not fit neatly into the 2-into-1 category. This can be illustrated by the known usefulness of the half-adder in addition and multiplier circuits (Figure 18.28).

18.6 CONCLUSIONS

In this chapter we have put forward the view that evolutionary algorithms together with the assemble-and-test methodology can be regarded as a discovery



18.28 FIGURE The half-adder subcircuit.

engine or creative machine for new designs. We studied this idea in the context of digital logic. We suggested that new principles may be able to be discovered by examining a series of evolved designs, in our case, for arithmetic logic circuits. We examined the concept of the space of all circuit representations but feel that similar ideas may well carry over to the general field of design. The human-designed algebras that form subsets of the space of all representations both for binary and multiple-valued systems are analogous to small "pools" of human principles, and by employing the blind evolutionary technique we may discover new principles. We also looked at the difficult problem of principle extraction from evolved data. We feel confident that the process of learning new principles from a blind evolutionary process is just a matter of time.

REFERENCES

Iba, H., M. Iwata, and T. Higuchi (1997). Machine Learning Approach to Gate-Level Evolvable Hardware. In T. Higuchi et al. (eds.), *Proceedings of the 1st Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES96), Lecture Notes in Computer Science 1259, Springer-Verlag, pp. 327–343.

Jain, A. K., R. J. Bolton, and M. H. Abd-El-Barr (1993). CMOS Mutliple-Valued Logic Design—Part 2: Function Realization. *IEEE Trans. on Circuits and Systems—I. Fundamental Theory and Applications* 40(8):515–522.

Kalganova, T., J. F. Miller, and T. C. Fogarty (1998). Some Aspects of an Evolvable Hardware Approach for Multiple-Valued Combinational Circuit Design. In M. Sipper et al. (eds.), *Proc. of the 2nd Int. Conf. on Evolvable Systems* (ICES 98), Lecture Notes in Computer Science 1478, Springer-Verlag, pp. 78–89.

Kalganova, T., J. F. Miller, and N. Lipnitskaya (1998). Multiple-Valued Combinational Circuits Synthesized Using Evolvable Hardware Approach. In *Proc. of the 7th Workshop on Post-Binary Ultra Large Scale Integration Systems* (ULSI'98), in association with ISMVL'98, Fukuoka, Japan, IEEE Press.

Kameyama, M., and T. Higuchi (1986). Synthesis of Optimal T-Gate Networks in Multiple-Valued Logic. In *Proc. of the 16th Int. Symposium on Multiple-Valued Logic,* IEEE Press, pp. 128–136.

Miller, J. F., and P. Thomson (1998a). Evolving Digital Electronic Circuits for Real-Valued Function Generation Using a Genetic Algorithm. In J. Koza et al. (eds.), *Genetic Programming: Proceedings of the Third Annual Conference,* Morgan Kaufmann, pp. 863–868.

Miller, J. F., and P. Thomson (1998b). Aspects of Digital Evolution: Evolvability and Architecture. In A. Eiben et al. (eds.), *Proceedings of the 5th Int. Conf. on Parallel Problem Solving from Nature* (PPSNV), Lecture Notes in Computer Science 1498, Springer-Verlag, pp. 927–936.

Miller, J. F., and P. Thomson (1998c). Aspects of Digital Evolution: Geometry and Learning. In M. Sipper et al. (eds.), *Proceedings of 2nd Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES98), Lecture Notes in Computer Science 1478, Springer-Verlag, pp. 25–35.

Miller, J. F., P. Thomson, and T. C. Fogarty (1997). Designing Electronic Circuits Using Evolutionary Algorithms. Arithmetic Circuits: A Case Study. In D. Quagliarella et al. (eds.), *Genetic Algorithms and Evolution Strategies in Engineering and Computer Science,* Wiley, pp. 105–131.

Sipper, M., E. Sanchez, D. Mange, M. Tomassini, A. Perez-Uribe, and A. Stauffer (1997). A Phylogenetic, Ontogenetic, and Epigenetic View of Bio-Inspired Hardware Systems. *IEEE Trans. on Evolutionary Computation* 1(1):83–97.

Thompson, A. (1997). An Evolved Circuit, Intrinsic in Silicon, Entwined with Physics. In T. Higuchi et al. (eds.), *Proceedings of the 1st Int. Conf. on Evolvable Systems: From Biology to Hardware* (ICES96), Lecture Notes in Computer Science 1259, Springer-Verlag, pp. 390–405.

19

CHAPTER

Discovering Novel Fighter Combat Maneuvers: Simulating Test Pilot Creativity

R. E. Smith The University of The West of England
B. A. Dike The Boeing Company
B. Ravichandran Scientific Systems
A. El-Fallah Scientific Systems
R. K. Mehra Scientific Systems

19.1 INTRODUCTION

New technologies for fighter aircraft are being developed continuously. Often, aircraft engineers can know a great deal about the aerodynamic performance of new fighter aircraft that exploit new technologies, even before a physical prototype is constructed or flown. Such aerodynamic knowledge is available from design principles, from computer simulation, and from wind tunnel experiments.

However, knowing the aerodynamic impact of a new technology is distinct from knowing how the plane will perform in combat. The mapping from aerodynamics to effective maneuvers in a new fighter aircraft is exceedingly complex. Discovering this mapping involves learning, adaptation, and the transference of knowledge from the maneuvers of other fighter aircraft. It is a creative task, usually accomplished in the cockpit, through the well-trained mind of an experienced pilot.

Evaluating the impact of new technologies on actual combat can provide vital feedback to designers, to customers, and to future pilots of the aircraft in question. However, due to the complex mapping discussed above, this feedback typically comes at a high price. While designers can use fundamental design principles (i.e., tight turning capacity is good) to shape their designs, often good maneuvers lie in odd parts of the aircraft performance space, and in the creativity and innovation of the pilot.

Therefore, the typical process would be to develop a new aircraft, construct a one-off prototype, and allow test pilots to experiment with the prototype, developing maneuvers in simulated combat. Clearly, the expense of such a prototype is substantial. Moreover, simulated combat with highly trained test pilots has a substantial price tag. Therefore, it would be desirable to discover the maneuver utility of new technologies, without a physical prototype.

There are several ways to approach this goal. One is to use man-in-the-loop combat simulation, where a real pilot flies a simulation of the new plane. This avoids the need for a physical prototype, but still includes the cost of trained fighter pilots, serving their role in simulated cockpits. Moreover, there is no guarantee that pilots will exhibit their instinctive, adaptive abilities on the ground, even in the most convincing, domed combat simulators. Certainly, the pilots would feel more at home with their senses embedded in the sights, sounds, and feelings of a real flight.

Another approach would be to analytically consider the mapping from aerodynamics to complex maneuvers. One could do this through combinations of differential game theory and optimal control. In general, such approaches involve modeling the combination of combat rules and aerodynamic performance in a mathematically tractable form. Of course, this is very difficult. Moreover, the process of making the combat situation tractable is likely to introduce structure into the problem, which will bias analysis toward rather obvious maneuvers as solutions. If the model is reconstructed to allow for new maneuver possibilities, it is likely that analytical techniques will find good maneuvers for these possibilities. However, the tractable modeling process is always likely to bias toward maneuvers that are actually *implicit* in the model. Such processes are unlikely to yield truly novel combat maneuvers.

A third approach is that pursued in the authors' past work, and in new work presented here. In this approach, an adaptive, machine learning system takes the place of the test pilot in simulated combat. This approach has several advantages:

- As in the analytical approach, this approach requires a model. However, in this case the model need only be accurate for purposes of combat simulation. It need not present a mathematically tractable form.
- Moreover, the approach is similar to that of man-in-the-loop simulation, except in this case the machine learning "pilot" has no bias dictated by past experiences with real aircraft, no prejudices against simulated combat, and no

tendency to tire of the simulated combat process after hundreds or thousands of engagements.

- Moreover, one overcomes the constraints of real-time simulation.

This chapter considers ongoing work in this area in terms of its unique character as a machine learning and adaptive systems problem. To recognize the difference between this problem and more typical machine learning problems, one must consider its ultimate goal. This work is directed at filling the role of test pilot in the generation of innovative, novel maneuvers. The work is not directed at online control. That is to say, the machine learning system is not intended to generate part of an algorithm for controlling a real fighter aircraft. Like the test pilot in simulated combat, the machine learning system can periodically fail, without worry that the associated combat failure will result in loss of hardware and personnel. In many ways, the machine learning system is even less constrained than the test pilot, in that it can experiment with maneuvers that would be dangerous in simulated fighter combat with real aircraft.

Moreover, the goal of this work is the process of innovation and novelty, rather than discovering optimality. It is difficult to define what is even meant by "the optimal combat maneuver." The system can have performance losses and gains as it learns and still be useful, if it discovers "interesting" maneuvers along the way. Although performance feedback is necessary to drive the adaptive process in this system, neither optimizing this feedback, nor monotonic increase in its value, are necessarily required for the system to have real-world utility.

The following sections introduce the details of this system and discuss how its differences in goals affect the techniques employed. The final sections discuss the implications of this work and suggest how machine innovation could serve as a new, interesting area for further research.

19.2 FIGHTER AIRCRAFT MANEUVERING

To understand motivations for the work presented here, and the results presented, one must have access to a few concepts of fighter aircraft maneuvering. This section provides an overview. For more comprehensive information, see Shaw (1998).

One critical tactical measure in air combat is target *aspect angle*, which is the angle between a target aircraft's line-of-sight and the line-of-sight of the attacking aircraft. If the target aspect angle is 0, then the target aircraft is pointed directly at the attacker. If the target aspect angle is 180, then the attacker is on the target's tail. Tactical advantage in close-in air combat can be measured as the difference between the target aspect angle and own-ship aspect angle, since the most desirable condition for the attacker is to point directly at the target while the target is pointed directly away from the attacker.

An important, new technological aspect of many modern fighter aircraft is the use of *post-stall technology* (PST). PST refers to systems such as thrust vectoring and advanced flight controls, which enable the pilot to fly at extremely high *angles-of-attack,* well beyond the normal stall limits of conventional aircraft. Angle-of-attack refers to the angle between the aircraft's velocity vector and the aircraft's nose, as measured in the geometric plane containing the aircraft's nose and vertical tail. Using PST flight modes, pilots have developed an entirely new class of combat maneuvers:

- The *Herbst maneuver,* which is the most well-known of PST maneuvers. In a Herbst maneuver, the aircraft quickly reverses direction through a combination of high angle-of-attack and rolling. The Herbst maneuver is named after Wolfgang Herbst, one of the original developers of PST.
- The *helicopter gun attack,* where the PST aircraft points its nose at a circling adversary while staying in the center of the adversary's turning circle.
- The *cobra maneuver,* which was first demonstrated by the Russian pilot Pougachev. In the cobra maneuver, the aircraft makes a very quick pitch-up from horizontal to 30 degrees past vertical. The airspeed of the aircraft slows dramatically as the plane continues its horizontal travel. The pilot then uses thrust vectoring to help pitch the aircraft's nose down and resume normal flight angles. This allows the aircraft to rapidly strip airspeed, causing a pursuing fighter to overshoot.
- The *PST hammerhead turn,* where the aircraft reverses direction by performing a maneuver resembling a backflip.

One typical attribute of PST tactics is *out-of-plane maneuvering,* where the attacking aircraft flies in a different maneuvering plane than the target. The *maneuvering plane* is determined by an aircraft's velocity vector and aerodynamic lift vector. A frequent characteristic of PST maneuvers is a continually changing maneuver plane.

PST maneuvers are primarily useful for air-to-air gun attacks because the gun is fixed in alignment with the aircraft's nose, requiring the aircraft to be pointed directly at the target. Air-to-air missile attacks do not require a similar degree of nose pointing because missiles turn after launch in the direction of the target.

Missiles are the preferred mode of attack for air combat. However, air-to-air gun attacks may be required in tactical situations if the target is inside the minimum missile range, or if the target employs countermeasures that defeat missiles.

Due to their complex dynamics, PST maneuvers are difficult to characterize without the use of 3D visualization tools. The primary visualization tool used in this study was Agile-Vu, a 3D graphics animation package developed by the U.S. Navy for analyzing aircraft flight trajectories. Agile-Vu provides the user with the ability to examine maneuvers from a variety of viewpoints and determine the degree of out-of-plane maneuvering.

In the experiments presented here, two aircraft are employed. One is the X-31 experimental fighter plane, which has significant PST characteristics. The other is an F-18, which has more "standard" fighter characteristics.

19.3 GENETICS-BASED MACHINE LEARNING

Although *genetic algorithms* (GAs) are most often used in optimization (Goldberg 1989; Holland 1992), a growing area of GA application is in machine learning. Optimization can be distinguished from learning in several important respects. In optimization, a system like the GA is given a fixed set of parameters to tune, and its only feedback is the utility of various parameter settings. In a machine learning problem, a system must interact with an environment that provides utility-related feedback, and feedback that indicates some time-varying state of the environment. In general, the system must find several high-utility sets of parameters that correspond to various state-feedback signals. One approach to machine learning problems is to convert them to optimization problems by assigning a full set of parameters for every possible environmental state, and simply optimizing. In general this approach is impractical. Instead, one would like for the learning system to generalize with sets of parameters that perform well across a range of environmental states. Because the appropriate degree of generalization is itself an unknown, the effective number of parameters needed is unknown as well. Moreover, it might be necessary to obtain these parameters gradually, in "chunks" that build toward a final, generalized solution. Thus, the problem is difficult to cast as one of typical optimization.

This work considers GAs in a machine learning context. A thorough review of genetics-based machine learning cannot be included here, for the sake of brevity. However, a concise review of *learning classifier systems* (LCSs), the primary genetics-based machine learning technique, is provided in the following section.

19.3.1 Learning Classifier Systems

Consider the following method for representing a state/action pair in a reinforcement learning problem (Grefenstette 1988): encode an environmental state in binary, and couple it to an action that can be taken in the environment, which is also encoded in binary. In other words, the string

0 1 1 0 / 0 1 0

represents one of 16 states and one of eight actions. This string can also be seen as a rule that says "IF in state 0 1 1 0, THEN take action 0 1 0." In an LCS, such a rule is called a *classifier.* One can easily associate a cost value, or other performance measures, with any given classifier.

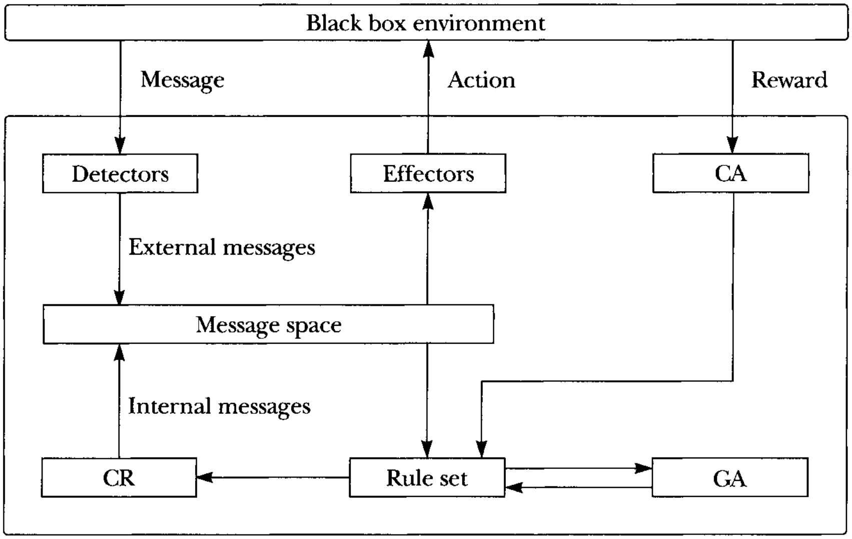
Now consider generalizing over actions by introducing a "don't care" character (#) into the state portion of a classifier. In other words, the string

# 1 1 # / 0 1 0

is a rule that says "IF in state 0 1 1 0 OR state 0 1 1 1 OR state 1 1 1 0 OR state 1 1 1 1, THEN take action 0 1 0." The introduction of this generality allows a system to represent clusters of states and associated actions. However, one can keep only a limited list of these "cluster templates" (which are also called "rules," or classifiers). The search problem involved is deciding which rules to keep, and thus design an effective, rule-dictated maneuver strategy. This is a search problem that naturally lends itself to the GA.

The GA is used in an LCS to locate a good set of generalizations (cluster templates, or generalized rules). Note that this problem requires a diverse set of cooperative classifiers to dictate a strategy. Thus, the LCS applied to this problem must *coevolve* an appropriate set of classifiers (Holland 1992). By using the GA to search for such strings, one can search for ways of clustering states together such that they can be assigned joint performance statistics (Holland et al. 1986; Holland 1992).

Figure 19.1 shows a diagram of the structure of an LCS. In a stimulus-response LCS (like the system used in this chapter), at each time step a message is posted by the system's detectors, representing the state of the environment. Classifiers are matched against this external message. From among the matching classifiers, some conflict resolution (CR) method is used to select a single classifier to act. That classifier's action is submitted to the environment, through the system's effectors. If reward (performance feedback) is received from the environment, it is distributed by a credit allocation (CA) system. Some LCSs also



19.1 FIGURE Structure of a learning classifier system.

allow for internal messages that represent communication between classifiers on successive time steps. However, this possibility is not considered here.

The GA is applied periodically after a specified number of operational cycles. Typically, a high-fitness portion of the classifier list serves as the GA population. Newly produced classifiers replace a low-fitness portion of the list.

Given the outline of the basic learning system structure, one must consider why an LCS is appropriate for discovering rules that dictate combat maneuver logic. The major motivation for this approach is the extensive theoretical and empirical evidence that GAs have robust adaptive ability in ill-conditioned search spaces (Holland and Reitman 1978; Goldberg 1989). Moreover, such systems allow for discovery of coadapted rules that dictate complex maneuvers. Finally, all these abilities only depend on direct, performance-only feedback, making the system ideal for learning from experience in simulation.

19.3.2 The LCS Used Here

The authors have extensive, ongoing experience with using LCSs for acquiring novel fighter aircraft maneuvers. This is a successful, ongoing project, satisfying real-world goals for industry, NASA, and the United States Air Force (Smith and Dike 1995; Smith et al. 2000).

By way of introduction to the system used here, consider the basic problem of one-versus-one fighter aircraft combat. Two aircraft start their engagement at some initial configuration and velocity in space. In our simulations, an engagement lasts for a prespecified amount of time (typically 30 seconds). An engagement is divided into discrete time instants (in our simulation, 1/10th second instants). At each instant, each "aircraft" must observe the state of the environment and make a decision as to its own action for that instant. A score for a given aircraft can be calculated at the end of an engagement by comparing that aircraft's probability of damaging its opponent to its own probability of being damaged.

Given this basic outline of the problem at hand, we will introduce details of the fighter aircraft LCS.

The Combat Simulation

The LCS interacts in simulated, one-versus-one combat, through the Air-to-Air System Performance Evaluation Model (AASPEM). AASPEM is a U.S. government computer simulation of air-to-air combat and is one of the standard models for this topic.

Detectors and Classifier Conditions

The conditions of the classifier are from the traditional {1, 0, #} alphabet, defined over the encoding of the environmental state shown in Figure 19.2.

Note that although this encoding is *crisp,* its coarseness and the open-ended bins sometimes used in the encoding have a linguistic character that is similar to that of a *fuzzy* encoding. That is, concepts like "high, low, and medium" are embodied in the encoding.

Effectors and Classifier Actions

The classifier actions directly fire effectors (there are no internal messages). Actions are from the traditional {1, 0, #} syntax and correspond to the coarse encoding shown in Figure 19.3.

Note that these action values are only *suggested* to the system that simulates the fighter combat. In many situations, it may be impossible for the aircraft to obtain the exact value suggested by the active classifier. Fortunately, the nature of the underlying fighter combat simulation returns an aerodynamically feasible setting of these action parameters, regardless of conditions. This feature of the system is discussed in more detail later.

	Bins (represented in binary, plus the # character)							
3 bits:	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Own aspect angle (degrees)	<45	45 to 90	90 to 35	135 to 180	180 to 215	215 to 270	270 to 315	315 to 360
Opponent aspect angle (degrees)	<45	45 to 90	90 to 135	135 to 180	180 to 215	215 to 270	270 to 315	315 to 360
2 bits:	**00**	**01**	**10**	**11**				
Range (1,000 feet)	<1	1 to 4.5	4.5 to 7.5	>7.5				
Speed (100 knots)	<2	2 to 3.5	3.5 to 4.8	>4.8				
Delta speed (100 knots)	<−0.5	−0.5 to 0.5	0.5 to 1	>1				
Altitude (1,000 feet)	<10	10 to 20	20 to 30	>30				
Delta altitude (1,000 feet)	<−2	−2 to 2	2 to 4	>4				
Climb angle	<−30	−30 to 30	30 to 60	>60				
Opponent climb angle	<−30	−30 to 30	30 to 60	>60				
Total: 20 bits								

19.2 FIGURE Encoding of environmental state in the fighter aircraft LCS.

	Action Values (suggested to AASPEM)							
3 bits:	**000**	**001**	**010**	**011**	**100**	**101**	**110**	**111**
Relative bank angle (degrees)	0	30	45	90	180	−30	−45	−90
Angle-of-attack (degrees)	0	10	20	30	40	50	60	70
2 bits:	**00**	**01**	**10**	**11**				
Speed (100 knots)	1	2	3.5	4.8				
Total: 8 bits								

19.3 FIGURE Action encoding in the fighter aircraft LCS.

As an example of the encoding, consider the classifier shown below:

01010010110111010010 / 11010010

This classifer decodes to

```
IF
      (90 > OwnAspectAngle > 35) AND
      (215 > OpponentAspectAngle > 180) AND
      (7.5 > Range > 4.5) AND
      (Speed > 4.8) AND
      (0.5 > DeltaSpeed -0.5) AND
      (Altitude > 30) AND
      (2 > DeltaAltitude > -2) AND
      (ClimbAngle < -30) AND
      (60 > OpponentClimbAngle> 30)
THEN ATTEMPT TO OBTAIN
      (RelativeBankAngle = -45) AND
      (AngleOfAttack = 40) AND
      (Speed = 3.5)
```

Match-and-Act

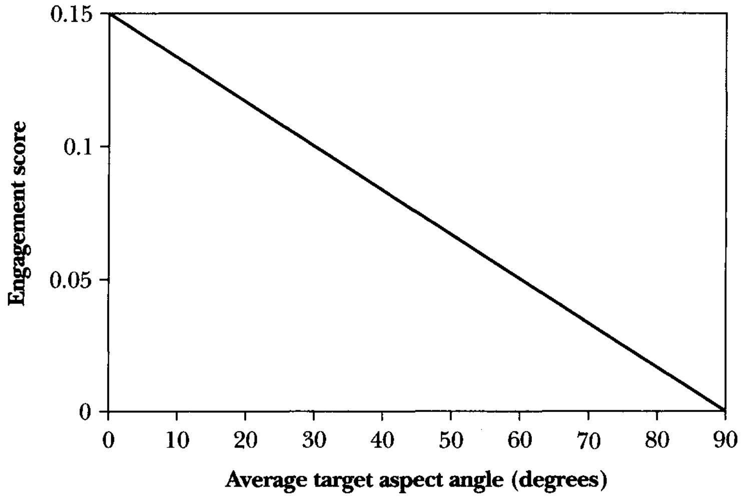
In our system if no classifiers are matched by the current message, a default action for straight, level flight is used. There is no "cover" operator (Wilson 1994).

Credit Allocation

At the end of an engagement, the "measure of effectiveness" score for the complete engagement is calculated (see below). This score is *assigned* as the fitness for *every* classifier that acted during the engagement (and to any duplicates of these classifiers). Note that this score *replaces* the score given by averaging the parent scores when the GA generated the rule. Thus, rules that do not fire simply "inherit" the averaged fitness of their GA parents (Smith, Dike, and Stegmann 1994).

Measures of Effectiveness

Our efforts have included an evaluation of different measures of effectiveness within the genetics-based machine learning system, to determine the relative



19.4 FIGURE Measure of effectiveness, which is used as an input to the GA fitness function.

sensitivity of the process. Initial candidate measures included exchange ratio, time on advantage, time to first kill, and other relevant variables.

The measure of effectiveness ultimately selected to feed back into the GA fitness function was based on the following steps. The base score was based on a linear function of average angular advantage (opponent target aspect angle minus own-ship target aspect angle) over the engagement, as shown in Figure 19.4.

To encourage maneuvers that might enable gun-firing opportunities, an additional score was added when the target was within 5 degrees of the aircraft's nose. A tax was applied to nonfiring classifiers to discourage the proliferation of parasite classifiers that contain elements of high-performance classifiers but have insufficient material for activation. All nonfiring classifiers that were identical to a firing classifier were reassigned the firing classifier's fitness.

GA Activation

The GA acts (over the entire population) at the end of each 30-second engagement. In some of our experiments, the *entire* classifier list is replaced each time the GA is applied. This has been surprisingly successful, despite the expected disruption of the classifier list. In recent experiments, we have used a generation gap of 0.5 (replacing half of the classifier population with the GA). This is still a substantially larger portion than are replaced in many LCSs.

A new, GA-created classifier is assigned a fitness that is the average of the fitness values of its "parent" classifiers. The GA employed tournament selection, with a tournament size ranging from 2 to 8. Other parameters are a crossover probability of 0.95, and a mutation rate of 0.02 per bit position. When a condition bit is selected for mutation, it is set to one of the three possible character values (1, 0, or #), with equal probability. Note that this actually yields an effective mutation probability of $(0.02)(2/3) = 0.0133$. Children rules replaced randomly selected rules in the population.

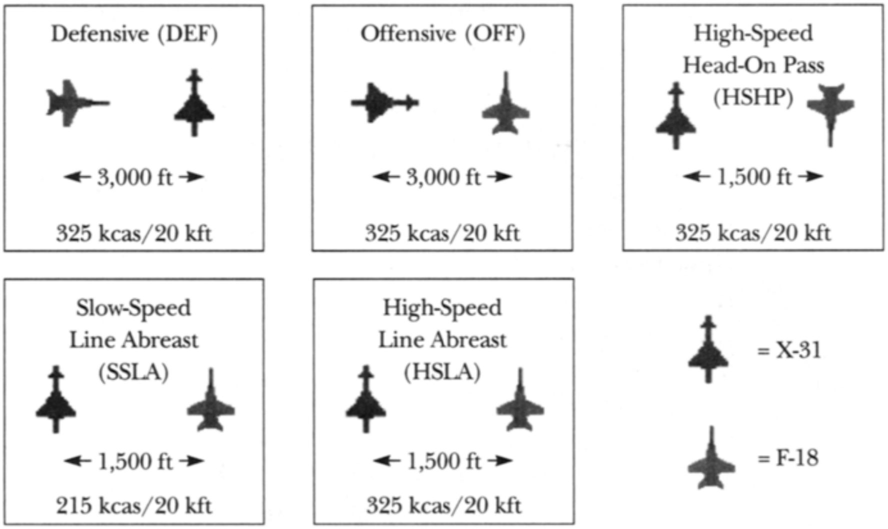
To resolve conflicts, the matching rule with the highest fitness/strength is selected to act *deterministically*.

Combat Simulation Starting Conditions

A two-tier approach was employed for defining run conditions and learning system parameters. First, a baseline matrix of initial positions, relative geometries, and energy states was identified in conjunction with NASA requirements. The primary source document for this step was the X-31 Project Pinball II Tactical Utility Summary, which contained results from piloted simulation engagements conducted in 1993 at Ottobrunn, Germany. Initial findings from the X-31 Tactical Utility Flight Test conducted at Dryden Flight Research Center were also used to compare with results from this project. The baseline starting condition test matrix, shown in Figure 19.5, was based on X-31 piloted simulation and flight test conditions and was tailored to the X-31 performance envelope, flight test range constraints, and other testing considerations.

Note that each of these represents a separate initial condition for a fighter combat simulation. Each learning run consists of repeated engagements, all starting from the same one of these conditions.

The first four start conditions—Defensive (DEF), Offensive (OFF), Slow-Speed Line Abreast (SSLA), and High-Speed Line Abreast (HSLA)—were derived directly from the Pinball II project. The fifth start condition, High-Speed Head-On Pass (HSHP), was added to the matrix to provide an additional geometry that would not exclusively result in a close turning fight. The opponent aircraft was an F/A-18. The baseline matrix formed a set of core conditions to generate X-31 tactics results for a balanced cross-section of tactically relevant conditions. The test conditions specified the initial geometries, X-31 and opponent speeds, altitudes, and ranges.


Defensive (DEF)
3,000 ft
325 kcas/20 kft
Offensive (OFF)
3,000 ft
325 kcas/20 kft
High-Speed Head-On Pass (HSHP)
1,500 ft
325 kcas/20 kft
Slow-Speed Line Abreast (SSLA)
1,500 ft
215 kcas/20 kft
High-Speed Line Abreast (HSLA)
1,500 ft
325 kcas/20 kft
= X-31
= F-18

19.5 FIGURE Baseline test matrix of initial conditions for the combat simulations.

19.4 "ONE-SIDED LEARNING" RESULTS

Results from the system outlined in Section 19.1 are documented elsewhere extensively (Smith and Dike 1995; Smith, Dike, and Stegmann 1994). However, results are reprinted here for clarification.

In our early efforts (Smith and Dike 1995), only one of the fighter aircraft employs the genetic learning system, while the other employs fixed, but reactive, standard combat maneuvers that are embedded in AASPEM. Simply stated, these fixed maneuvers instruct the opponent to execute the fastest possible turn to point its own nose at the LCS-controlled aircraft, while attempting to match its altitude.

In this "one-sided learning" configuration, the system has found a variety of novel fighter aircraft maneuvers that were positively evaluated by actual fighter test pilots. One maneuever discovered by the LCS is shown in Color Plate 16. The aircraft on the left is following a fixed but reactive strategy, while the aircraft on the right is following a strategy evolved by the LCS. Moreover, the system discovers maneuvers from a variety of well-recognized fighter aircraft strategies

(Smith and Dike 1995). For instance, one result was a variant of the well-documented Herbst maneuver.

19.5 "TWO-SIDED LEARNING" RESULTS

In more recent work with the fighter combat LCS, we have allowed both opponents to adapt under the action of the GA (Smith et al. 2000). This ongoing effort complicates the fighter combat problem and the interpretation of simulation results. These complexities are best seen from the perspective of extant literature on two-player games.

In many ways, the one-versus-one fighter combat scenario is like a continuous version of the *iterated prisoners' dilemma* (IPD) (Axelrod 1984; Luce and Raiffa 1989). In the IPD, the game takes place in distinct rounds, in which each player can take one of two actions: cooperate or defect. If both players cooperate, both players get a moderate reward. If one player defects while the other cooperates, the former gets a large reward, and the latter gets a large punishment (negative reward). If both defect, both get an even larger punishment.

This is similar to the situation in the one-versus-one fighter combat scenario. Each aircraft has a choice of attacking the other (analogous to defecting), or evading the other (analogous to cooperating). This analogy is useful, since IPD has been studied a great deal, both with static and adaptive players. However, it is important to note that the current work may be one of the first explorations of a real-world, continuous time analog of IPD with two adaptive players.

The IPD literature for coadaptive players shows us several interesting behaviors. Each of these is an artifact of the *Red Queen effect*, so-called because the Red Queen in Alice in Wonderland states that in her world you must keep running just to stand still (Floriano and Nolfi 1997). In an analogous way, the performance of each player in the two-sided learning problem is relative to that of its opponent. In other words, when one player adapts and the other uses a static strategy (as in our previous work), the performance of the adaptive player is absolute with respect to its opponent. However, when both players are adaptive, the performance ceases to have an absolute meaning. Instead, its meaning is only relative to the state of its current opponent. This is an important effect that must be considered in the interpretation of our current results.

Also because of the Red Queen effect, the dynamic system created by two players has several possible attractors, including the following;

- *Fixed points:* where each player adapts a static strategy and is willing to cope with a (possibly inferior) steady state

- *Periodic behavior:* where each player cycles through a range of strategies, sometimes losing and sometimes winning
- *Chaotic behavior:* where each player visits a random series of strategies, with no eventual escalation of tactics
- *Arms races:* Where each player continuously ramps up the sophistication of its strategies

The last is clearly the behavior we want our simulations to encourage. Our current results have (qualitatively) shown promise in this area (i.e., we have seen an escalation of strategies between the two aircraft).

A number of approaches to two-sided learning have been considered. In each approach, a "run" consists of 300 simulated combat engagements. Approaches employed include the following:

- *Alternate freeze learning (ALT):* In each run one side is learning while the other is frozen (not altering its rule base). The frozen player uses the population (rules) obtained from the previous run, in which it was learning. The learning player starts each new run with a random set of rules. In other words, rules are learned "from scratch" against the rules learned in the last cycle by the other player.
- *Alternate freeze learning with memory (MEM):* This learning scheme can be viewed as an extended version of the ALT learning. At the end of each run, the results of the 300 engagements are scanned to obtain the highest measure of effectiveness. The rules from the highest scoring engagement are used for the frozen strategy in the next run. Furthermore, these rules are memorized and are added to the population in the upcoming learning sequence runs. Thus, the system has memory of its previously learned behavior.
- *Parallel learning (PAR):* Genetic learning goes on continuously for both players.

Each of the three two-sided algorithms was tested and evaluated on all five Pinball cases (see Figure 19.5). In all experiments a population of maneuvers was generated in an iterative process, beginning with an initial population of random classifiers. The maneuver population was successively improved using AASPEM air combat engagements to evaluate the maneuver.

A typical two-sided learning result (and one that indicates a learning "arms race") is shown in Color Plate 17. Color Plate 17 (top) shows the "best" maneuver (in terms of the measure of effectiveness discussed in Section 19.3) discovered for the red player. In this maneuver, the red player (an F-18) fails to have an advantage for more than the last half of the engagement due to agile maneuvers performed by the blue player (an X-31, a somewhat superior aircraft). The X-31 executes a nose dive maneuver followed by a Herbst-type maneuver allowing it to gain an advantage. Although red does not gain advantage, it is interesting to note its evasive behavior in reaction to blue's maneuver.

Color Plate 17 (bottom) is the blue player's "best" maneuver (once again, in terms of the measure of effectiveness), which is evolved after the result shown in Color Plate 17 (top). It is likely that learning from the red player's evasion resulted in the improved behavior seen here. This illustrates the advantage of two-sided learning. However, note that the meaning of "best," in terms of a static measure of effectiveness, is not entirely clear in these simulations, due to the red queen effect. Therefore, further investigation of the two-sided learning system is needed. However, the two-sided learning system is already yielding valuable maneuver information, in relationship to the system's overall goals.

19.6 DIFFERENCES IN GOALS AND TECHNIQUES

We believe much of the success of the system presented above is due to the particular goal at which the system is directed. From the outset of the fighter aircraft LCS project, the goal has not been to directly *control* fighter aircraft. Instead, the focus has been on the discovery of novel maneuvers for their own sake. The utility of this goal is revealed when one considers the complexity of the aircraft combat task (Shaw 1998).

In a general sense, one can understand the aircraft's *aerodynamic* characteristics before an actual prototype is generated. However, given the complexity of the aircraft combat task, one cannot map this directly to maneuvers that are advantageous. Therefore, test pilots and extensive combat trials (mock or real) are generally needed to discover the innovative maneuvers that will convert raw aerodynamics into combat success.

Therefore, the goal of our system is simply the discovery of novel maneuvers, not optimality. There is, in general, no such thing as an optimal maneuver, and discovery of such a maneuver is not the system's goal. Discovering many *successful* combat maneuvers in this complex task has quantifiable advantages, including

- feedback to aircraft designers with regard to the combat advantages of various aerodynamic characteristics
- feedback to fighter pilots on the use of various aircraft characteristics
- feedback to customers on the advantages of a given aircraft's characteristics

These advantages have real-world value. Moreover, they are advantages that could transfer to any number of other tasks where the discovery of novel strategies is difficult and where new strategies have intrinsic values.

19.6.1 Implications of This Goal

In general, the goal of novelty discovery shifts the perception of what techniques are advantageous. For instance, the pursuit of Q-values that yield a Bellman optimal control strategy seems less critical in this setting. While it is true that one wants to pursue high-utility maneuvers, it is not expected that an "optimal" maneuver can be found. Moreover, enumerating all possible high-utility maneuvers is probably not possible. Finding classifier-based generalizations over the *entire* payoff landscape is probably not possible as well.

Most importantly, maintaining any *consistent* control strategy is not of great importance in this task. The frequent tendency of LCSs to have periodic failures is not a particular concern. Consider the typical, but previously unpublished result from the fighter aircraft system shown in Figure 19.6. In this typical run, only one aircraft is using the LCS, while the other uses standard combat maneuver logic, as in the run shown in Color Plate 16.

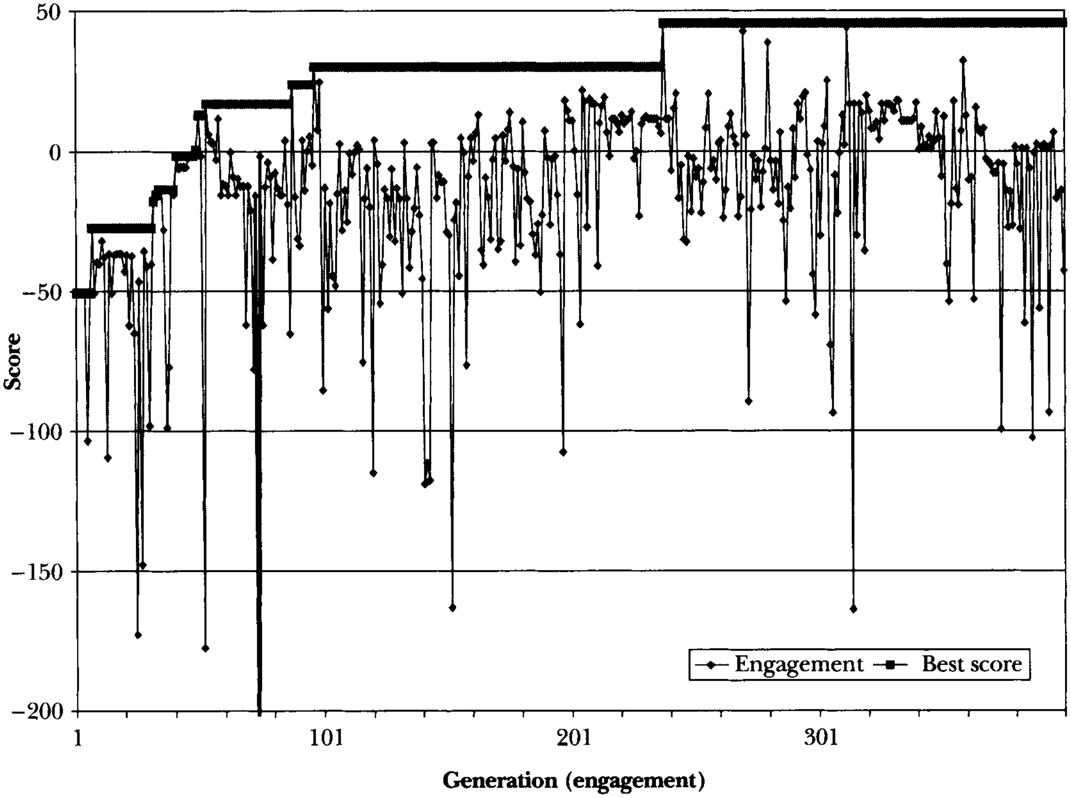
This figure shows that the fighter aircraft LCS frequently loses rules and suffers temporary losses of performance. However, the system moves to improved performance (and new maneuvers). Thus, it accomplishes the goal of *discovery*, but not necessarily the goal of control.

19.7 CONCLUSIONS

The work presented here is a part of an ongoing effort. The LCS presented is an artifact of several years of incremental design, and its performance is likely to benefit from more recent developments in LCS technology. However, the system continues to provide useful results in a complex, real-world domain.

The main key to this system's success is in the nature of the application itself. This task is one where discovering novelty has quantifiable value to designers, pilots, and customers (rather than online control). Discovering similar application domains, where using genetics-based machine innovation has a real-world value, is an important area for future efforts.

On a more technical level, there are features of the current fighter aircraft LCS that deserve consideration in the design of other LCSs.



19.6 FIGURE Engagement score for the LCS aircraft versus generation number.

One should consider the "smoothed" interpretation of actions in the fighter combat LCS. The authors strongly suspect that this is a reason for the LCSs success on this task. Many of the action representations used in LCS research are "brittle"; that is, rule deletion, and other forms of classifier set disruption, can easily cause less-than-graceful failure. In the fighter aircraft task, the actions of rules are implicitly interpreted in a linguistic, aerodynamically appropriate manner. This makes each classifier less sensitive to changes in the composition of the classifier set, which are likely to occur, given the action of the GA.

One should also consider the fighter aircraft LCS credit allocation scheme. Given the efficacy and popularity of *Q*-learning and related algorithms (e.g., SARSA, which is essentially the implicit bucket brigade), it is easy to overlook that there are well-founded techniques that do not follow their form. In particular, the Monte Carlo RL methods are just as firmly related to reinforcement learning theory. In fact, Sutton and Barto (1998) indicate that such methods may be more appropriate than Q-learning and related methods when one faces a non-Markovian task.

However, such methods are only appropriate for tasks with clearly defined episodes. The fighter aircraft task is clearly episodic, since engagements take place in a predefined time window. When one directs an LCS at an episodic task, epochal credit assignment schemes, like that used in the fighter aircraft LCS, may be of higher utility than methods based on Q-learning, SARSA, or the bucket brigade.

Like the Q-learning-based methods used in LCSs, epochal schemes can only approximate the associated reinforcement learning techniques, since such techniques are typically based on table lookup. Given that the focus of the LCS is on finding generalized rules, associated reinforcement learning techniques, be they epochal or not, must be adapted to evolving, generalized rule sets. XCS (Wilson 1995) seems to have found an appropriate adaptation of Q-learning for an LCS. Finding an appropriate adaptation of well-founded, epochal reinforcement learning schemes to LCS use is an area worthy of future research.

However, when considering the technical details of reinforcement learning control (that is, obtaining optimal strategies in reinforcement learning problems), one should not overlook the utility of the LCS approach for generating novel, innovated approaches to problems. Such goals may not fall within the rubric of strict optimality, but in many domains (like the fighter aircraft task), such open-ended discovery can have a real-world, hard-cash value. The applicability of the LCS approach to such creative tasks deserves further consideration.

ACKNOWLEDGMENTS

The authors gratefully acknowledge that this work is sponsored by the United States Air Force (Air Force F33657–97-C-2035 and Air Force F33657–98-C-2045). The authors also gratefully acknowledge the support provided by NASA for the early phases of this project, under grant NAS2–13994.

REFERENCES

Axelrod, R. (1984). *The Evolution of Cooperation.* Basic Books.

Floriano, D., and S. Nolfi (1997). God Save the Red Queen!: Competition in Co-evolutionary Robotics. In *Proceedings of the Second International Conference on Genetic Programming,* pp. 398–406.

Goldberg, D. E. (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning.* Addison-Wesley.

Grefenstette, J. J. (1988). Credit Assignment in Rule Discovery Systems Based on Genetic Algorithms. *Machine Learning* 3:225–246.

Holland, J. H. (1992). *Adaptation in Natural and Artificial Systems.* MIT Press.

Holland, J. H., K. J. Holyoak, R. E. Nisbett, and P. R. Thagard (1986). *Induction: Processes of Inference, Learning, and Discovery.* MIT Press.

Holland, J. H., and J. S. Reitman (1978). Cognitive Systems Based on Adaptive Algorithms. In D. A. Waterman and F. Hayes-Roth (eds.), *Pattern Directed Inference Systems,* Academic Press.

Luce, R. D., and H. Raiffa (1989). *Games and Decisions.* Dover Publications.

Shaw, R. L. (1998). *Fighter Combat: Tactics and Maneuvering.* United States Naval Institute Press.

Smith, R. E. and B. A. Dike (1995). Learning Novel Fighter Combat Maneuver Rules via Genetic Algorithms. *International Journal of Expert Systems* 8(3):247–276.

Smith, R. E., B. A. Dike, R. K. Mehra, B. Ravichandran, and A. El-Fallah (2000). Classifier Systems in Combat: Two-Sided Learning of Maneuvers for Advanced Fighter Aircraft. In *Computer Methods in Applied Mechanics and Engineering,* Elsevier.

Smith, R. E., B. A. Dike, and S. A. Stegmann (1994). Inheritance in Genetic Algorithms. In *Proceedings of the ACM 1995 Symposium on Applied Computing,* ACM Press, pp. 345–350.

Sutton, R. S., and A. G. Barto (1998). *Reinforcement Learning: An Introduction.* MIT Press.

Wilson, S. W. (1994). ZCS: A Zeroth-Level Classifier System. *Evolutionary Computation* 2(1):1–18.

Wilson, S. W. (1995). Classifier Fitness Based on Accuracy. *Evolutionary Computation* 3(1):149–176.

20 CHAPTER Innovative Antenna Design Using Genetic Algorithms

Derek S. Linden Linden Innovation Research LLC

20.1 INTRODUCTION

Since F. Braun created the first wire antenna in 1898, a plethora of antennas have been designed, using reflectors, horns, metal patches, wires, and the like. These antennas are designed in large part using an inductive process. In general, an engineer finds an existing design that may have the desired electromagnetic characteristics. The engineer then works with exact or approximate equations, if any exist, to determine the design's proper dimensions and parameters and uses a simulator to predict its performance. If the performance is not good enough, the engineer redesigns and resimulates the antenna, using intuition to determine which parameters to change to improve performance. This design cycle has produced, over time, the many different antenna designs with varied characteristics, but it is time-consuming and unlikely to produce truly optimal results. It requires that the engineer be familiar with the many different designs that exist and have enough experience and expertise so that an acceptable solution can be reached in a reasonable amount of time. It also tends to limit the design to relatively simple structures.

With the advent of ever-faster high-speed computers, it has been possible to analyze more and more complex wire structures in reasonable times and also to optimize wire antennas using computer-aided design technologies. In general, however, the basic topology of the wire antenna is still predetermined, and the individual parameters that constitute that particular configuration are optimized. While the computer allows for greater design complexity, the nature of most optimization techniques requires the engineer to use conventional methods to provide an initial guess fairly close to the final answer. But there are too many variables that interact in complex ways in even simple designs for a person to optimize effectively. Even years of experience may not result in useful intuition about certain aspects of electromagnetic problems.

This design cycle limits the types of designs that are tried to those that have an intuitive logic about them. Symmetry is often present, and structures are kept relatively simple to allow for easier understanding and analysis. Nearly all of the designs produced by engineers have the characteristic of "making sense" when one looks at them, their circuit schematics, or parameterized models. They *look* like they should work. Such is not the case with the antennas designed by the GA processes discussed here. Though they do work in simulation and in actual measurement, it does not appear logical or reasonable that they should work. They are quite different from those a rational human designer would ever think of, let alone try to design.

GAs are being applied to many different antenna designs by many different researchers. In Rahmat-Samii and Michielssen's recent book on the topic of genetic algorithms in electromagnetic optimization (Rahmat-Samii and Michielssen 1999), seven of the chapters concern antenna design. GAs and other evolutionary computation techniques are able to show such progress over other methods of antenna design for several reasons:

- Simulators are fast and accurate, requiring only seconds to produce accurate results for interesting designs.
- Search spaces are highly rugged and resistant to other forms of numerical and hands-on optimization, yet finding good designs is important to industry.
- Antenna principles, which are a subset of electromagnetics and founded on Maxwell's equations, are extremely difficult to understand and grasp intuitively.
- Antennas have been studied since the late 19th century, and thus there is a large body of knowledge that can help the engineer to constrain the optimization properly.

Most antenna optimizations begin with a conventional design and the GA finds the optimal parameters based on desired conventional characteristics. For instance, an inherently high-directivity design like the Yagi-Uda antenna may be optimized for maximum gain (these terms are defined in the next section). This approach is certainly useful, since even conventional problems are difficult to optimize with most other methods, and the resulting optimized designs will often be better than any found previously.

However, of greater interest here is applying conventional designs to unconventional applications, where the GA has enough degrees of freedom to significantly change the mode of operation of the antenna to suit the new application. Two such examples will be described here, which apply Yagi-Uda antennas in unconventional roles. In this manner, the GA begins to show some (though only a small portion) of its innovative capability.

Of still greater interest is the ability of the GA to find antenna designs where very little information is given. Constraining the GA to a conventional design provides it with an amount of engineering knowledge that may help to produce better solutions, but limits what the GA can find to the inherent characteristics of the design. However, GAs are able to find good antennas with very little constraint—certainly nothing as constraining as a conventional human-created antenna design. Several examples will be shown of what the GA can do with so few constraints. This ability to go beyond human-created designs will become more and more significant as time progresses. As the world goes increasingly wireless, there is a growing number of antenna problems without good solutions. The tracking of hospital patients, biomedical research, wideband data communications, remote sensing, integration of antennas within electronic devices, and so on are all demanding antennas that meet their needs. Meeting them rapidly and effectively will require new approaches to antenna design because the current methods and conventional designs are too limited to keep pace with the rising demand. What the GA may be able to provide is an automated design system that can explore areas of antenna design previously unsearchable and solve antenna design problems unhindered by the limits of human intuition and experience. This need is the motivation behind the research presented here.

20.2 ANTENNA BASICS

This section provides a brief tutorial on antenna design concepts that will help the reader to understand the designs described in later sections. There are many kinds of antenna classes, such as reflector antennas (e.g., dish antennas), phased array antennas (consisting of multiple regularly spaced elements), wire antennas, horn antennas, and microstrip and patch antennas. Each of these classes use different structures and exploit different properties of electromagnetic waves. An antenna is a wire antenna if it is constructed from conductors that are much longer than their width. Wire antennas can be fed a signal, or excited/driven, many different ways. A wire antenna can be fed most simply with a coaxial line at its base, like a car antenna. It is also possible to feed a wire antenna using a balanced, two-wire line, as in a TV antenna where there are two wires connected to the "rabbit ears," or using even more complex schemes.

The wires do not always have to be connected. Unconnected elements are called parasitics and act as passive antennas that receive radiation from surrounding elements and reradiate them in different directions. Such parasitics can produce excellent results, as will be shown in Section 20.3.

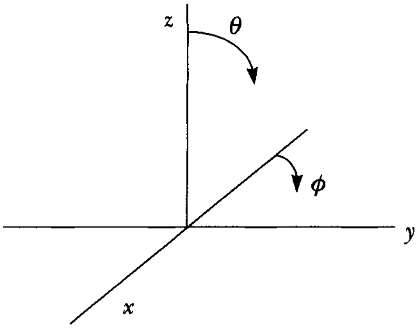
The research presented here is limited to the class of wire antennas. There are a few reasons for choosing wire antennas over other classes: First, the simulator of choice, the Numerical Electromagnetics Code, Version 2 (NEC2), as discussed later in this section, is a fast, accurate, widely used, public-domain wire antenna simulator. Second, wire antennas are easy and inexpensive to build and test, and thus there is a cost-effective way to validate the results of an optimization. Third, wire antennas are a particularly flexible class—they can be used to solve many different types of design problems: high or low gain, broad or narrow band. Other classes, like horns and reflectors, are much more restricted in the types of problems that they can solve. Following are some antenna terms with their definitions.

At its simplest, a *ground plane* is a large, flat piece of metal underneath the antenna. It is often used in conjunction with a wire antenna. It acts as a mirror for the antenna above it, and therefore changes the antenna gain pattern. A ground plane can decrease the required height and/or simplify the construction of the wire antenna. The hood or roof of a car acts as a ground plane, and antennas that will be affixed to such places need to be designed for use with one.

Directivity and *gain* are two related qualities in antenna design. Directivity is the ratio of power density being transmitted by an antenna in a particular direction to the average power density being transmitted in all directions. The gain is the directivity multiplied by the ratio of power radiated to power input. Gain takes into account all losses such as loss due to resistance in the antenna, which converts some of the input power into heat, and loss due to mismatch between the transmitter/receiver and the antenna. When the losses are considered to be zero, as in this chapter, the directivity and gain are equal.

Gain is usually expressed in *decibels* (dB), which relates to a ratio of power or power densities by the following expression: $\mathrm{dB} = 10\log_{10}(P_1/P_2)$. In the case of gain, P_2 is the power density of an isotropic radiator that transmits power equally in all directions. The abbreviation dBi refers to gain compared with an isotropic radiator. However, the "i" is sometimes left off and is understood from context.

A *gain pattern* or *antenna pattern* plots gain magnitude versus angle, showing the proportion of power an antenna transmits in a particular direction. For 2D



20.1 FIGURE θ and ϕ on a 3D axis system. Arrows begin where NEC2 defines 0 degrees for θ and ϕ.

antennas, or antennas symmetric in the third dimension, this angle is simply the elevation angle θ. In 3D, there are two angles that specify a direction: θ and the azimuth ϕ. Figure 20.1 shows these angles on a set of axes. An antenna is considered to be *directive* if its gain pattern is heavily weighted in one direction.

An antenna's *beamwidth* refers to the useful angle span of the so-called main lobe or beam. This lobe usually has the highest gain in the pattern and is what is of interest to optimize. In a uniform gain pattern, there is only one lobe, but in a directive pattern where the beamwidth is desired to cover only a certain angle span, there can exist other lobes. These other lobes are called *sidelobes,* and usually the designer seeks to minimize them.

Bandwidth is the useful range of frequencies for an antenna and is usually desired to be as large as possible. It is given in percent, which is the ratio of the useful frequency span over the nominal operating frequency. For an antenna operating at 2GHz, a bandwidth of 3% would mean it would operate over a 60 MHz range, from 1.97 GHz to 2.03 GHz. For the applications covered in this chapter, a bandwidth of 2–5% is usually the minimum desired and means an antenna is very sensitive to exact dimensions and can be used in only specialized applications; 10% bandwidth is usually considered relatively broad; and 20% is considered to be quite broad.

Voltage Standing Wave Ratio (VSWR) is a way to quantify the match between an antenna and a device connected to it. A standing wave is created when there is a mismatch in this connection, which prevents power from flowing to and from the antenna. If the standing wave is large, implying a high VSWR, there is a significant mismatch. If it is low, the match is good. A VSWR of 3.0 or less is considered adequate for many low-power applications, while a VSWR less than 1.5 or 2.0 is desired if power considerations are important. A VSWR of 1.0 is a perfect match, and it can never be less than 1. VSWR is easy to measure, and since it is a common parameter specified by antenna designers, it is often an important quantity to optimize.

Polarization refers to the orientation of electromagnetic waves. Electromagnetic waves are composed of two components: an E-field (electric field) component, which is a sinusoidal wave that exists in one plane, and an H-field (magnetic field) component that exists at right-angles to the E-field following the right-hand rule ($\mathbf{E} \times \mathbf{H}$ = a vector in the direction of propagation). Thus, the wave is asymmetric and has a definite orientation. Since the H-field is constrained by the E-field in a propagating wave, we will discuss just the E-field. In a wave of constant overall magnitude, the E-field magnitude can actually be a time-varying quantity. If one looks at a wave propagating past a fixed point in space, the E-field can oscillate back and forth in a single orientation, giving *linear polarization,* or it can actually be rotating in a circle as its x and y components oscillate back and forth out of phase, giving *circular polarization.* It can also take an orientation in between, giving *elliptical polarization.* Antennas are polarization-sensitive: An antenna that is optimal for picking up linear polarized signals will miss half of the energy of a wave that is circularly polarized and all of a wave that is cross-polarized (linearly polarized at a right angle to the antenna). An antenna that is left-hand circularly polarized (the E-field moves in a circle to the left) will miss a right-hand circularly polarized wave completely. In addition, for ground-to-satellite communications, circular polarization is very helpful because it minimizes the effect of polarization distortion that occurs as a signal travels through the ionosphere.

The size of a wire antenna can vary greatly. If a wire antenna is desired to be directive, it will generally need to be several wavelengths in size. However, a gain pattern that is to be uniform will require a much smaller antenna.

There are many electromagnetic simulators that exist for wire antennas. One particularly suited to the task of creating wire antennas is the Numerical Electromagnetics Code, Version 2 (NEC2) (Burke and Poggio 1981). This code was used exclusively in this research. NEC2 has a simple file interface for input and output that makes it ideal for using with an optimizer. Its input file allows for nearly arbitrary wire structures—a feature that is key in allowing a GA to produce innovative designs that no rational designer would have chosen. NEC2 is also in the public domain, so obtaining and modifying the source code is cost-free and easy, as is copying the simulator between machines.

Perhaps most importantly, NEC2 has a long track record of being accurate and reliable. The NEC2 code was produced in the early 1980s and has been used to simulate antenna structures for many years. It has shown itself to be in very good agreement with actual measurements, and thus one has confidence that simulated performance will have validity. This feature is crucial, since the optimization process is only useful if the simulator is accurate in modeling reality.

Now that antennas and their conventional design process have been covered, following is a discussion of innovative wire antennas that have been created through GAs, beginning with unconventional applications of the Yagi-Uda antenna.

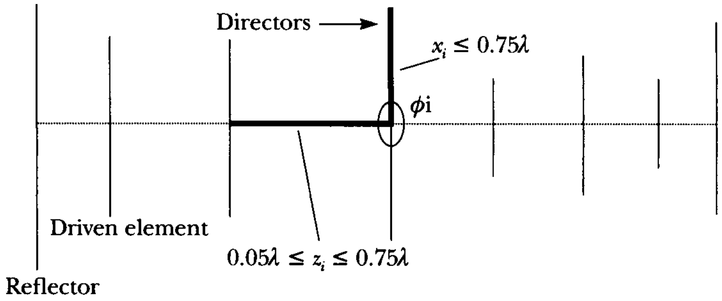
20.3 CONVENTIONAL DESIGNS AND UNCONVENTIONAL APPLICATIONS: THE YAGI-UDA ANTENNA

As shown in Figure 20.2, the Yagi-Uda antenna is a series of parallel wires, first proposed by Prof. Yagi and his student S. Uda in the late 1920s. One element is driven, one element is behind the driven element and is called the *reflector*, and all other elements are called *directors*. The highest gain can be achieved along the axis and on the side with the directors. The reflector acts like a small ground plane, allowing power that would otherwise be sent backward to be reflected forward.

The conventional Yagi design includes geometry variables of length for each element, spacing between elements, and the diameter of the wire. Thus, with N elements, there are N length variables, $N - 1$ spacing variables, and one wire diameter variable, giving $2N$ variables total. The driven element is driven from its center.

One other variable beyond that of the conventional Yagi that produced good results is shown in Figure 20.2. It is designated as ϕ_i and is a degree of freedom relating to the rotation of the element out of the plane of the antenna. Its effect will be discussed later in this section.

An unconventional application of this antenna is described by Altshuler and Linden (1997) and involves designing a special feed for the Arecibo 305-meter spherical reflector in Puerto Rico (Avruch et al. 1995). The antenna was to be used to search for primeval hydrogen having a redshift of approximately 5. Neutral hydrogen line emission is at a frequency of 1420 MHz; thus the frequency region of interest was about 235 MHz. Preliminary studies indicated that the band from 219 to 251 MHz was of the greatest interest, particularly from 223 to 243 MHz. The most important design goal was for the feed to have sidelobes at least 25 dB down from the main beam gain in the region from $70° < \phi < 290°$, due to the interference that came from surrounding radio and TV towers. Of lesser importance was that the E-plane (the plane parallel to the plane of the antenna) and H-plane (perpendicular to the E-plane) beamwidths be about 50°. The



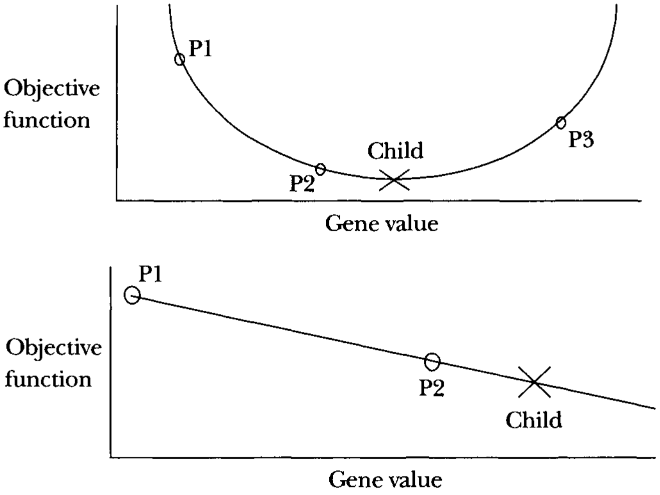
20.2 FIGURE The Yagi-Uda antenna.

VSWR was desired to be under 3.0, and the gain was to be as high as possible, limited by the wide beamwidth. The feed would be mounted over a 1.17 meter square ground plane—that is, a ground plane only 0.92λ in size. The antenna also needed to have a single polarization so that two of them could be operated at cross-polarity (i.e., each would have a polarization exactly opposite the other), and thus could be used to discriminate between the randomly polarized hydrogen signal and the deliberately polarized signals from the surrounding radio and TV towers. This arrangement would work best if the antennas were two-dimensional, so that they could be collocated at the same position at right angles.

Since there did not seem to be any conventional antenna that would meet the above specifications, it was decided to use a GA to optimize a Yagi type structure for this unconventional application. Yagi antennas are usually used for high-gain, narrowband applications. The desired bandwidth and beamwidth were very large for this kind of antenna, and the sidelobe requirements were very difficult to meet. However, a standard Yagi antenna is two-dimensional and therefore able to meet the polarization and collocation requirements.

The GA used to optimize this problem had a real-valued chromosome and used Adewuya's method of crossover (Adewuya 1996). Adewuya's method consists of a sequence of crossover methods applied to real genes. First, quadratic crossover is applied, where the child's gene is taken from a predicted minimum of a quadratic curve fit using three parents. If quadratic crossover fails, heuristic crossover is applied, which pulls the child's gene from a range predicted to be better than two parent's genes. See Figure 20.3 for a graphical representation of what happens in these two methods.

If both quadratic and heuristic crossover fail, the child's gene is one of the parent's genes taken at random. This process is applied gene by gene to create a



20.3 FIGURE Quadratic and heuristic crossover. The objective function is to be minimized in these examples.

new child. See Adewuya (1996) and Linden (1997a, 1997b) for a more complete explanation and comparisons with other methods. This method has been found to be particularly powerful in electromagnetics and mechanical engineering.

The number of wires was specified to be 14. The variables were the length of each element (14 were required), each constrained to be symmetric and between 0 and 1.5λ; the spacing between each set of two elements (13 were required), constrained to be between 0.05λ and 0.75λ (the total boom length was allowed to vary); and the diameter of the wire, constrained to be 2, 3, 4, 5, or 6 mm. This wide latitude in parameters allowed the GA to explore very unconventional areas of the Yagi search space.

Note that of the total 28 variables, 27 of them were continuous, real-numbered parameters, making this a natural problem for a real-valued chromosome. The discrete variable—wire diameter—used a real-valued gene, but it was discretized using truncation so the GA would only use one of the allowed values. Doing so is usually not recommended for the type of crossover techniques used, but the problem was insensitive to this parameter, and it did not affect results adversely.

The GA was a steady-state type with a 30% overlap from generation to generation, 175 individuals in the population, and a 0.6% mutation rate. Mutation was Gaussian—mutated genes were pulled from a distribution with a mean equal to the unmutated gene, and a standard deviation of 0.1 of the full gene range. These values were shown to be optimal by experimentation in other Yagi antenna optimizations.

The objective function used to optimize the design was

$$\text{Fitness} = -G_L + C_1 \cdot SLL^2 + \sum_i (C_2 V_i)$$

where G_L is the lowest of all frequencies at 0°, SLL is the highest sidelobe level within $70° < \phi < 290°$ for all frequencies, and where V_i is VSWR at the ith frequency. The summation is over all frequencies:

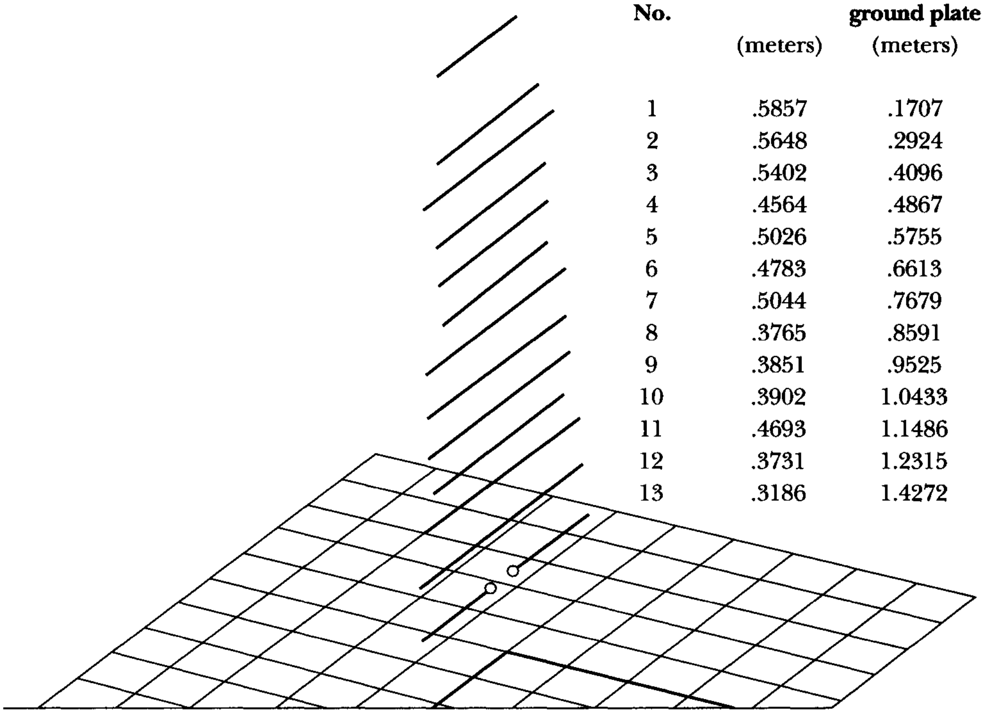
$$C_1 = \begin{cases} 0 & SLL > 25\,\text{dB} \\ 1 & SLL < 25\,\text{dB} \end{cases}$$

$$C_2 = \begin{cases} 0.1 & V < 3.0 \\ 1 & V > 3.0 \end{cases}$$

The optimization involved minimizing this function. Since beamwidth was of secondary importance, it was not included specifically in the cost function. Instead, beamwidth was implicitly defined by where the objective function began to search for sidelobes (at ±70°), and by the maximization of the gain. The antenna was simulated at 243 and 223 MHz, near the edges of the desired frequency band. It was assumed that if the antenna performed satisfactorily at these frequencies it would probably be acceptable over the rest of the band.

Although the feed was over a finite ground plane, it was decided to initially use a conventional reflector element in the design since modeling a finite ground plane using NEC adds a prohibitive amount of computer time. After an optimal configuration was obtained, a thorough computational analysis was conducted for the whole frequency band from 219 to 251 MHz at increments of 2 MHz to ensure that the antenna was truly broad in bandwidth. It was performed initially for the Yagi with only a reflector element and then repeated with a 1.17-meter ground plane.

As expected, the GA produced an antenna that approached the above requirements, though its configuration was quite unconventional for a Yagi antenna. It differed from a conventional Yagi in that the director elements were very closely spaced, its overall length was much less than a typical Yagi with the same number of elements, and its wire lengths varied haphazardly. The genetic Yagi had 13 elements (plus the ground plane) with a boom length of only 1.11λ. The directors varied in length in a seemingly random fashion from about 0.25λ

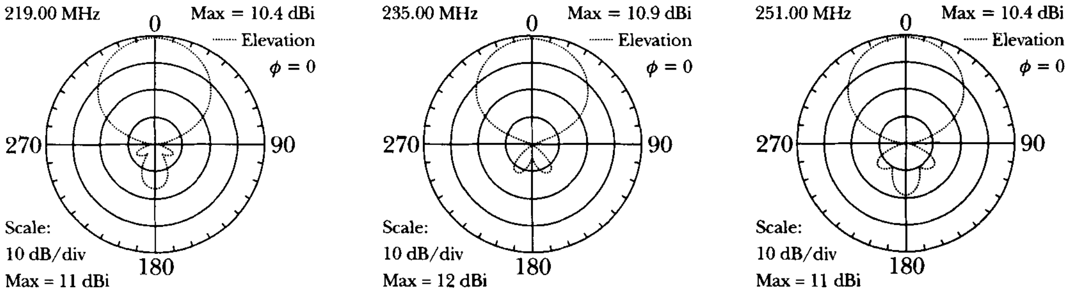


Element No.	**Length** (meters)	**Distance from ground plate** (meters)
1	.5857	.1707
2	.5648	.2924
3	.5402	.4096
4	.4564	.4867
5	.5026	.5755
6	.4783	.6613
7	.5044	.7679
8	.3765	.8591
9	.3851	.9525
10	.3902	1.0433
11	.4693	1.1486
12	.3731	1.2315
13	.3186	1.4272

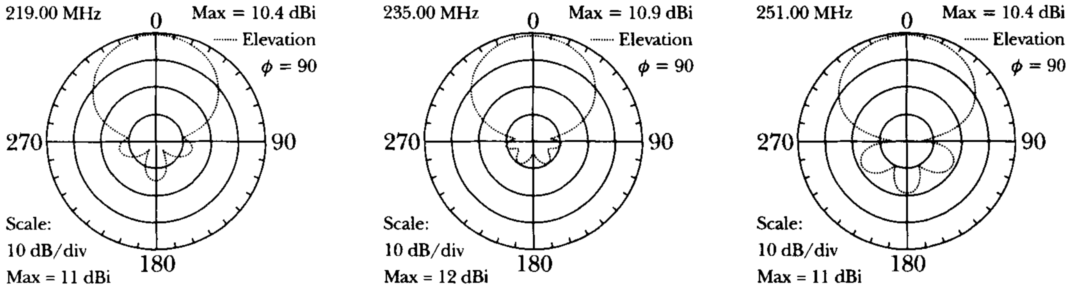
20.4 FIGURE Genetic Yagi feed for the Arecibo radio telescope. From Altshuler and Linden (1997).

to 0.4λ, with an average spacing of less than 0.1λ, as shown in Figure 20.4. A conventional 14-element Yagi has a boom length about 3 times as long, with directors that are about 0.4λ in length and 0.35λ in spacing, and the lengths become slightly shorter and the spacings slightly larger the greater the distance from the driven element.

The performance of this Yagi was computed at 2 MHz increments over the band from 219 to 251 MHz, a bandwidth of 13.6%. Figure 20.5 shows the E-plane patterns and H-plane patterns for the genetic Yagi over a finite ground plane at the same frequencies. It is seen that the sidelobe levels for both planes are more than 25 dB lower than the gain at 0° from 223 to 243 MHz, the most important part of the band, and are more than 20 dB lower over the rest of the frequency band. The E- and H-plane half-power beamwidths range from 51° to 55° and 64° to 69°, respectively, slightly larger than desired but certainly acceptable. The VSWRs are less than 3.0 from 227 to 245 MHz, though they are higher at the



E-Plane



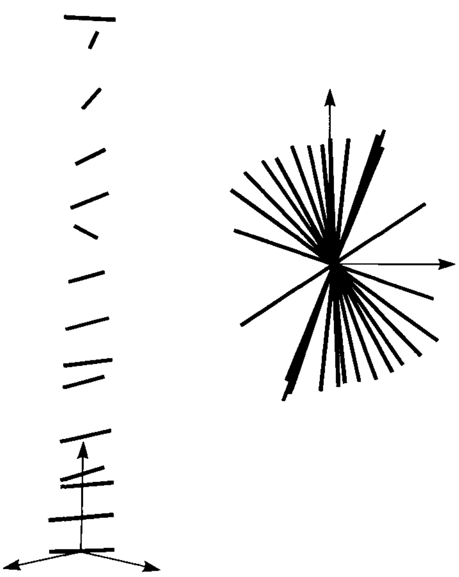
H-Plane

20.5 FIGURE Computed gain patterns of Yagi over a ground plane at 219, 235, and 251 MHz. From Altshuler and Linden (1997).

ends of the frequency band. The antenna gain ranged from 10.4 to 11.0 dB over the frequency band. This gain is approximately 1 dB lower than that for a Yagi that is optimized for maximum gain.

The antenna was fabricated to a one-sixth scale and the E-plane patterns and VSWR were measured. The computed and measured patterns had reasonable agreement. The measured VSWRs were less than 3.0 over most of the band and had a maximum value of 3.7 near the ends. The measured gains were slightly less than 10 dB; however, if the reflection losses are taken into account, the corrected values for a matched antenna approach the computed gains. For more information about this antenna, see Altshuler and Linden (1997, 1999).

This antenna shows the power of the GA to mold conventional designs into new forms to solve difficult problems. Naturally, it is important to allow the GA sufficient latitude in the design parameters to change the design from its traditional form to something new. However, it can also be useful to allow new degrees of freedom as well as wide latitude, which can lead to even more innovative and interesting results.



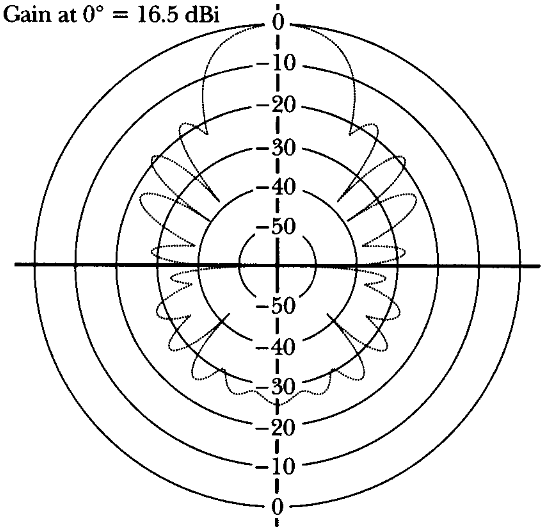
20.6 FIGURE Rotated Yagi—side and top views. Boom length was 5.16λ. From Linden (1997a).

For example, the traditional Yagi design consists solely of planar elements, which constrains it to be linearly polarized. It was this feature that made the Yagi attractive for the Arecibo feed problem. However, if one wishes to communicate with satellites through the ionosphere, circular polarization is desired. If one adds the ϕ_i variable as shown in Figure 20.2, the Yagi can take on other polarizations, perhaps even circular polarization if the elements are arranged properly. Is the GA sufficiently powerful to use this degree of freedom to make a new type of antenna—a circularly polarized Yagi?

As described in Linden (1997a), a GA was run with the same parameters as above, only with the additional degrees of freedom encoded in the chromosome. The objective function (to be minimized) had two terms: Fitness = −Gain + VSWR, but the gain took into account circular polarization, so that it lost value if the antenna was not circularly polarized.

This optimization produced a remarkable result. The antenna that the GA created was indeed circularly polarized. Figures 20.6 and 20.7 show its design and its gain pattern.

Usually, this kind of pattern is achieved using a helix antenna, essentially a single wire that looks like a spring with carefully measured spacing and diameter. It is difficult to make and maintain because the wire must be wound according to



20.7 FIGURE Normalized circular polarization gain pattern for rotated Yagi, in the plane of the antenna ($\phi = 0°$). Each circle is 10 dB lower than the one that encircles it. Circular polarization losses have been taken into account. From Linden (1997a).

exacting standards. But the GA allows the engineer to use a Yagi, which is easier to make, for the same purpose. Though the Yagi must also be made to exact specifications, it is much easier to make short wires the correct size and place them at the correct spacing than to wind a single long wire to the same level of precision.

Thus, not only can the Yagi be applied in unconventional ways using GA optimization, it can use extra degrees of freedom to turn a conventional design into an unconventional one. This design is a type of hybrid, in which human engineering and innovation through GA combine to form a unique type of antenna. However, it is not always necessary to specify a design at all, in which case the GA is truly the sole inventor of an antenna, as the next section will show.

20.4 UNCONVENTIONAL DESIGNS AND CONVENTIONAL APPLICATIONS: CROOKED-WIRE AND TREELIKE GENETIC ANTENNAS

The application of interest in this section is fairly conventional: satellite-to-ground communications using omnidirectional antennas. Antennas for this application, intended for use on cars or handsets, must be cheap, robust, and have as uniform a gain pattern across the hemisphere as possible for a right-hand circularly polarized signal, excluding low elevations less than 10° above the horizon, where multipath problems will arise. (*Multipath* refers to the reception of a signal from more than one path, such as receiving a signal from direct line-of-sight and from a reflection off the ground. When the multipath signals do not arrive in phase and at the same time, as generally happens, problems arise such as the ghosting seen on televisions.) This antenna is not trivial to design, and several conventional designs, such as the quadrafilar helix, have emerged to solve this problem to varying degrees. These antennas tend to be expensive and narrowband, and they often require a signal to be fed to the antenna in two places with a precise difference in phase to set up the circular wave. There is thus considerable room for improvement in the state of the art.

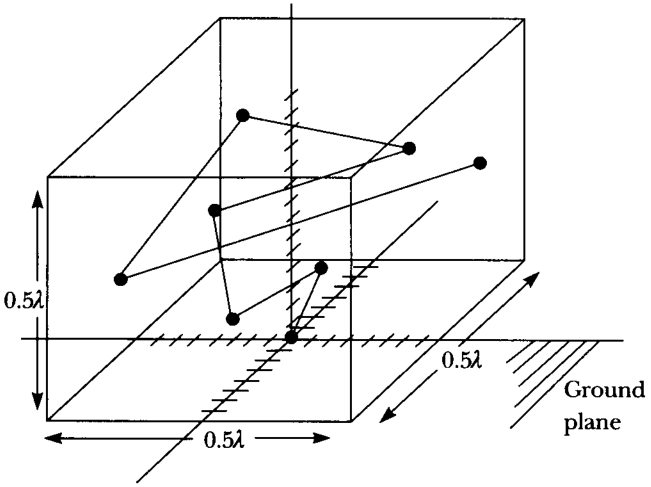
20.4.1 The Crooked-Wire Genetic Antenna

There are several qualities that one might desire an antenna to have, which can be turned into general constraints on the design. For instance, one might desire a single feed point at the base of the antenna for simplicity and low cost. In addition, it is helpful to have such an antenna over a ground plane. The antenna is expected to be relatively small because near-hemispherical coverage is desired, so it would make sense to constrain the search space to a fairly small volume, for instance, a cube half a wavelength on a side, with the antenna's base located in the center of the bottom face. Doing so will increase the average speed of simulation while having little impact on results.

It would also be convenient if the antenna were a connected series of straight wires so that there are no precision bends, floating parasitics, or branch points required, for ease in fabrication. five, six, seven, and eight straight wire segments connected in series were thus chosen for investigation. (Preliminary results showed the seven-wire antenna performed slightly better than the others, so seven wires will be used from here on.) A depiction of the search space that incorporates these constraints is shown in Figure 20.8.

The GA has performed well up to this point, but it has always had a preexisting design to use as a template for its work. Can a GA really produce a good antenna with its only engineering knowledge coming from constraints based on convenience?

To begin encoding this problem for a GA, the location of the nodes defining the start and end points of the wire segments were mapped into a chromosome. Since each node requires three coordinates, the 21 parameters were placed



20.8 FIGURE Crooked-wire genetic antenna search space for seven wires.

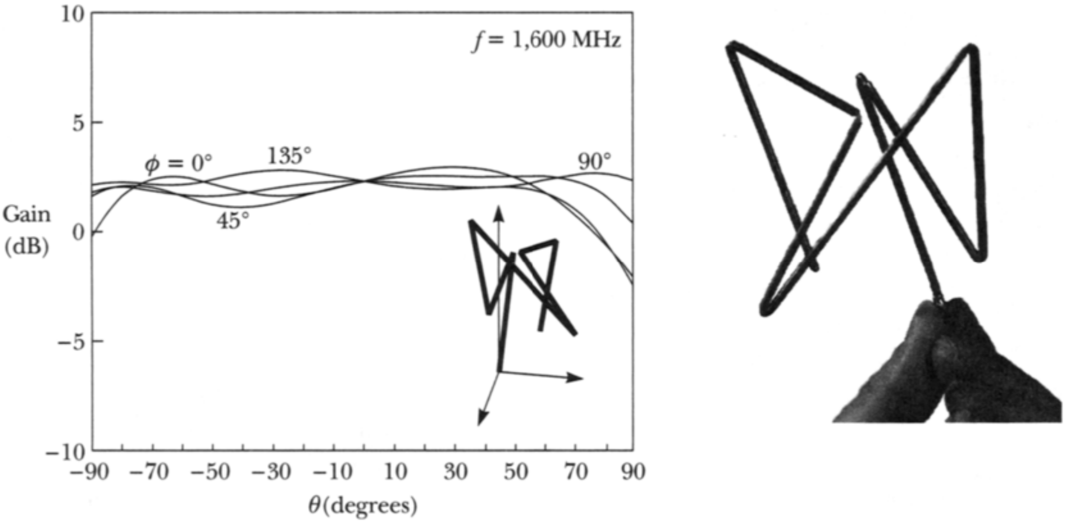
serially into a chromosome, that is, Point1-X, Point1-Y, Point1-Z, Point2-X, and so on, each value encoded into five binary bits. Five bits, corresponding to 32 levels, was chosen because the accuracy of fabrication was not expected to be better than this resolution (3 mm at 1,600 MHz). Thus, the whole chromosome required 105 bits.

The cost function was then determined for this antenna. The goal was to obtain right-hand circular polarization 10° above the horizon at a frequency of 1,600 MHz. A good measure of that desired performance can be found in the sum of the squares of the deviation of all calculated gains from the mean. In equation form:

$$\text{Fitness} = \sum_{\theta,\phi} (\text{Gain}(\theta,\phi) - \text{mean Gain})^2$$

The GA's goal was to minimize this fitness. For its first attempt at finding a seven-wire antenna, the steady-state GA had a population of 500 chromosomes, 50% overlap from generation to generation, and a 1% mutation rate. It also used one-point crossover, which was allowed to occur between any two bits in the chromosome with equal probability.

After several hours, the GA converged on a seven-wire configuration with a highly unusual shape, as shown in the inset and the photograph in Figure 20.9.



(a) (b)

20.9 FIGURE (a) Crooked-wire genetic antenna radiation pattern and diagram along with (b) photograph of the actual antenna. From Linden (1997a).

While legal and valid from NEC2's point of view, this shape was so unusual and its simulated performance so good that great care was taken to ensure its validity, including building and testing it (Linden and Altshuler 1996).

The computed radiation patterns of the antenna over an infinite ground plane are shown in Figure 20.9 for azimuth angles of 0°, 45°, 90°, and 135° at a frequency of 1,600 MHz. This pattern varies by less than 4 dB for angles over 10° above the horizon—excellent performance, especially since the antenna is so inexpensive and simple to build. (Simple is a relative term, of course, for it does not look all that simple in the figure. However, it was possible to fabricate this antenna by hand using very simple tools, while a conventional design would be tremendously difficult to manufacture by hand.)

Although this antenna was only designed to operate at a single frequency, its performance was also investigated for the range of 1,300 to 1,900 MHz, and it was found to have bandwidth of over 30%, which is excellent for a circularly polarized antenna having near hemispherical coverage.

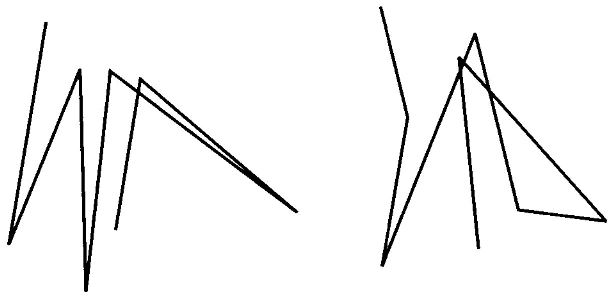
The antenna was built and measured for its radiation properties. There was about a 6 dB variation in the field above an elevation angle of 10° as compared to the computed variation of about 4 dB. This small discrepancy exists because the measurements were made over a 1.2 m × 1.2 m ground plane, while the computations were performed for an infinite ground plane. Patterns measured over the frequency range from 1,300 to 1,900 MHz also compared well with the computed patterns.

After this spectacular result, many other GAs, both binary and real-valued, were run for these requirements, and the results were never the same (Linden 1997a). Two more antennas optimized with the same constraints, chromosome, and fitness function are shown in Figure 20.10. Though they have nearly the same performance, they are quite dissimilar in shape from each other and from the antenna above. From these and other runs it is apparent that this search space is highly multimodal, with many minima that give similar performance (Linden 1997a).

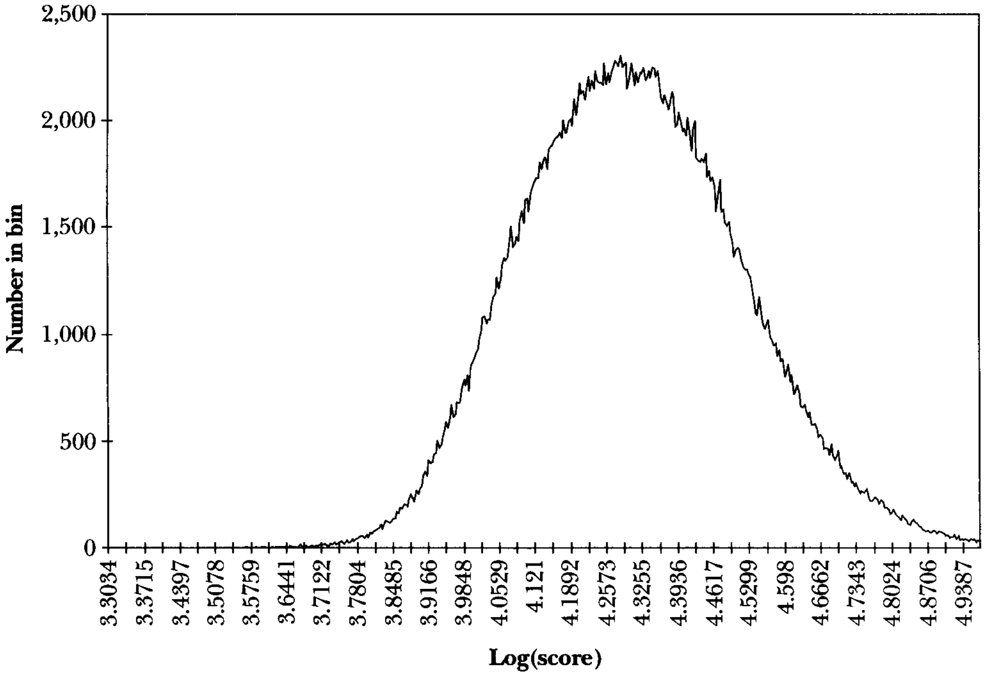
It is of interest to determine how important the GA search method is to the discovery/design of such antennas. If the search space has a high density of good solutions, it may be possible to use a simpler technique, such as a simple stochastic sampling of the search space coupled with a simple stochastic or gradient hill-climber, to find acceptable solutions. To explore this possibility, more than 360,000 randomly generated seven-wire designs were evaluated using NEC2. The distribution of the log of the scores was found to be close to normal, with a mean of 4.337 and a standard deviation of 0.208 (Figure 20.11). Thus, the average score is about 22,000. About 95% of the scores—all scores within ±2 standard deviations from the mean—lie between 8,300 and 57,000. Thus, the chances of randomly finding an acceptable solution is 1.6×10^{-15}, implying that the expected number of runs to achieve an acceptable solution is 6.3×10^{14}. An acceptable score is considered to be on the order of 800 or less in this case (meaning a 0.8 dB average variation over the hemisphere.) For comparison, the score of the first seven-wire genetic antenna design described above was about 330, implying a 0.5 dB variation over the hemisphere. It is not surprising to see this distribution, for it requires special design or luck to create an antenna that is particularly bad or particularly good. However it is rather surprising to see the tightness of the distribution about 22,000, and to see that it is very close to a normal distribution with only a slight skew.

This antenna is the subject of ongoing research, and it has been applied to several other problems, to include very small antennas (Altshuler 1999), uniform gain for low elevation angles over a lossy ground (Sandlin 1997), and adaptation of an antenna to its environment for omnidirectional and high-gain applications (Linden 2000). (For more information on this antenna, see also Rahmat-Samii and Michielssen 1999; Altshuler and Linden 1997, 1999; Linden and Altshuler 1996.)

Because of the revolutionary nature of this antenna design process, a patent has been awarded (Altshuler and Linden 1998). This patent, which appears to be the first of its kind (Weiss 1999), applies not only to the process of creating a



20.10 FIGURE Two more crooked-wire antennas with nearly identical performance.



20.11 FIGURE Distribution of randomly selected crooked-wire genetic antennas.

new antenna with no previously known underlying theory of operation, but to all the antennas created by this process. Thus, this patent applies to the crooked-wire genetic antenna of this section and the antennas in the next section.

Though patented by people, the antennas are the innovation of the GA, for it does not have much useful configuration input from the designer, and any constraints that it does have are made more for convenience than for antenna design soundness. Many antennas are named for their inventor, so it makes sense to call these genetic antennas, for the GA process is the inventive force behind them.

While the crooked-wire genetic antenna has few constraints already, it is possible to relax even more of them, as will be seen with the treelike genetic antenna.

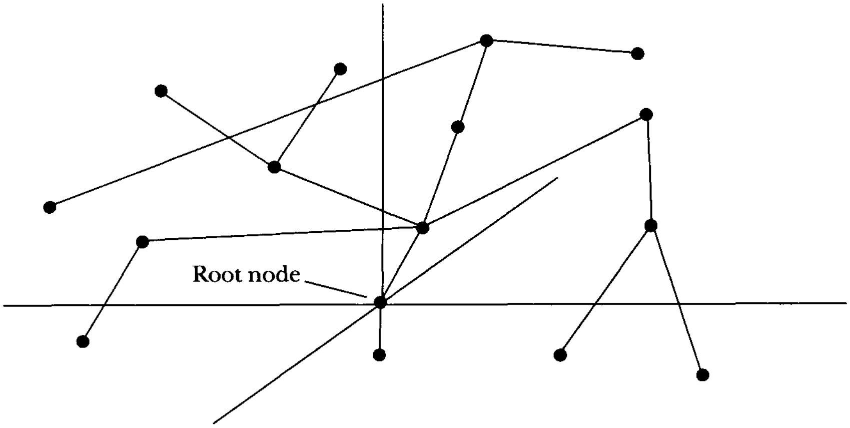
20.4.2 Treelike Genetic Antennas

This type of antenna uses a completely different kind of chromosome to generate an antenna: a tree chromosome. Treelike genetic antennas (or tree antennas) are thus very different from all previous antenna structures. The tree chromosome has 3D coordinates in its nodes. Wires for the NEC2 input file are drawn between parents and children in the tree. The root node is at the origin. Each wire can be a designated size, allowing for only angular data to reside in the nodes, or wires can be of different distances. The chromosome is actually a fractal structure, able to branch, and very large trees with very complex connections can result (Figure 20.12). This freedom has led to some surprising results so far.

For the runs presented here, each wire in the tree was limited to one segment in length (0.075λ), and each node contained relative coordinate information from its parent (i.e., its direction relative to its parent node). The program that writes the NEC2 input file takes this relative information and converts it to absolute coordinates, as NEC2 requires.

The initializer routine for the tree constructs a tree with a certain maximum number of children per node and a certain maximum depth. It places random 3D vector information in each node, where the vector is 0.075λ long and points in a random direction. The mating process includes both node and subtree swapping, and mutation included subtree destruction, node replacement and swapping, and subtree swapping.

Figures 20.13 and 20.14 show two resulting antennas. They were applied to the Arecibo feed problem described above, so they were optimized for a 60° beamwidth centered at 0° θ, low sidelobes, and linear polarization. To facilitate



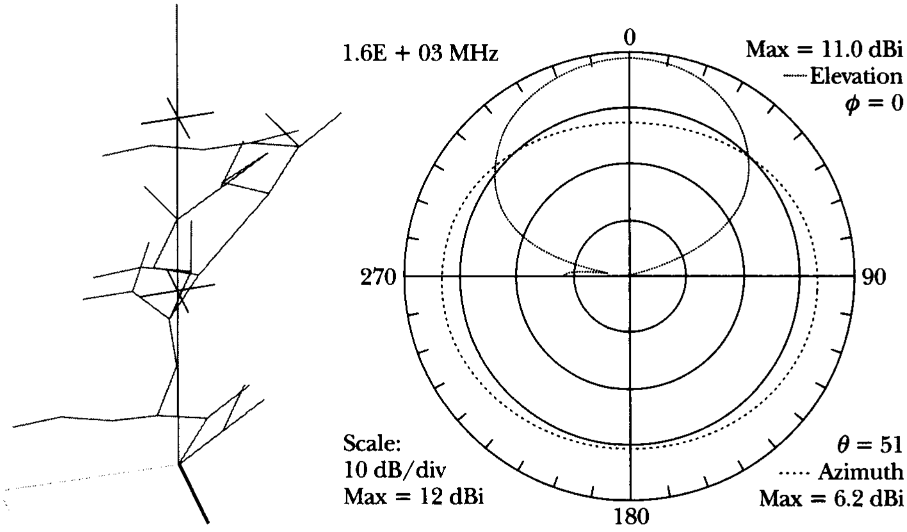
20.12 FIGURE Tree antenna design space.

the linear polarization, they were limited in movement in the *y* plane, producing almost flat antennas. They were allowed some movement in this third dimension to allow fewer wire crossings. They were both generated over a ground plane, and the vectors of the nodes were only allowed to point in the $+z$ direction to prohibit wires crossing below the ground plane (an illegal condition for NEC2).

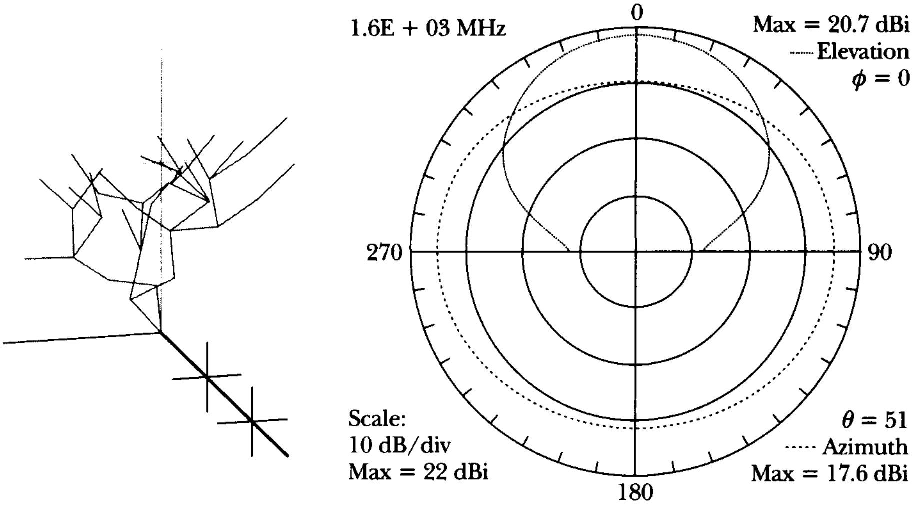
Notice that they seem to have very regular-looking patterns for such unusual shapes. In spite of the large number of wires present in each design, the pattern seems well-behaved. However, though they are interesting designs, they are not realistic antennas.

First, it is almost certain that these two antennas violate the assumption in NEC2 that wires are not touching except at segment boundaries. This probably is the cause of the high absolute gains that are shown in Figures 20.13 and 20.14 (i.e., Max = 20.7 dBi for elevation and Max = 17.6 dBi for azimuth), which are not realistic. On the other hand, when offending wires are removed that are certainly violating NEC2 assumptions, the patterns remain regular and close to those shown above. Experience has also indicated that the antenna directivity pattern is somewhat robust to such violations. Thus, the directivity indicated may not be too far from reality and shows that perhaps a well-behaved pattern is possible with these chaotic-looking structures.

Second, the antennas are not realistic because they are too cumbersome to fabricate and are likely to be too fragile for a realistic environment. The wires in this design are very thin, to help avoid illegal crossings, so there is little structural support. This problem can certainly be overcome by embedding the antenna in


1.6E + 03 MHz
0
Max = 11.0 dBi
Elevation
$\phi = 0$
270
90
Scale:
10 dB/div
Max = 12 dBi
180
$\theta = 51$
Azimuth
Max = 6.2 dBi

20.13 FIGURE Tree antenna 1.


1.6E + 03 MHz
0
Max = 20.7 dBi
Elevation
$\phi = 0$
270
90
Scale:
10 dB/div
Max = 22 dBi
180
$\theta = 51$
Azimuth
Max = 17.6 dBi

20.14 FIGURE Tree antenna 2.

a dielectric, but it would also be preferred to make them with fewer wires, to avoid unnecessary expense.

These preliminary results, then, seem to hold promise, but much more work is needed to make these antennas useful. Restrictions probably should be placed on the structures to ensure they do not violate the assumptions of the simulator and that they are able to be easily assembled. More work also needs to be done to discover the types of problems best solved with these structures, and the best parameters for the GA that optimizes them. However, these antennas show that a GA can design antenna structures that have the desired simulated characteristics even if they are given very little constraint.

20.5 CONCLUSIONS

Each of the antennas described above demonstrated a different quality of GAs as applied to wire antenna design. The Yagi antenna optimized for the Arecibo feed problem shows how the GA can change conventional designs, using them with unusual parameters, to solve unconventional problems. The rotated-element Yagi antenna shows how the GA can also incorporate unconventional degrees of freedom into conventional designs to give a hybrid design that is a joint effort between the GA and the engineer. The crooked-wire and treelike genetic antennas show the raw power of the GA to find not just an optimized design for an application, but to create a new design with minimal help from the engineer.

It may seem extremely surprising that a GA can autonomously find such amazing antennas. However, consider that many people have optimized antennas without any knowledge of electromagnetic theory through the adjustment of their TV's "rabbit ears" antenna. What is used is a rather stochastic local-search technique, based on feedback from the quality of reception, sometimes even involving haphazard pieces of aluminum foil, to find the best reception. It is usually unknown if the television viewer has found the best possible reception, but the process is stopped once reception has been found that is "good enough" or appears unlikely to improve. The antenna configuration and characteristics are usually quite different from the original V-shaped design, but it works, at least while the surrounding conditions remain constant. In addition, many different configurations will often give the same performance.

Similarly, the GA uses feedback from the antenna simulator to search, somewhat more effectively, the large search space of antenna configurations to find one that is acceptable. As with most complex engineering problems, it is very difficult to tell if the GA has found the best antenna, but often that is not as important as having an acceptable solution. And as in the case of the television antenna, many different antenna configurations may give similar performance, depending on the problem. So while the results shown in this chapter are indeed remarkable, they are not unreasonable, given the nature of antenna design.

In summary, then, GAs are able to optimize wire antennas for many diverse and difficult applications. The inherent power of the GA to not only optimize conventional designs, but to create them virtually on its own, makes it an ideal method of automated design for wire antennas.

REFERENCES

Adewuya, A. (1996). *New Methods in Genetic Search with Real-valued Chromosomes.* Master's thesis, Mechanical Engineering Department, MIT.

Altshuler, E. E. (1999). Small Wire Antenna Design Using a Genetic Algorithm. In the *Proceedings of the URSI General Assembly.*

Altshuler, E. E., and D. S. Linden (1997). Wire Antenna Designs Using a Genetic Algorithm. *IEEE Antenna & Propagation Society Magazine* 39:33–43.

Altshuler, E. E. and D. S. Linden (1998). *A Process for the Design of Antennas Using Genetic Algorithms.* U.S. patent 5,719,794. Issued February 17, 1998.

Altshuler, E. E., and D. S. Linden (1999). Design of Wire Antennas Using Genetic Algorithms. In Y. Rahmat-Samii and E. Michielssen (eds.), *Electromagnetic Optimization by Genetic Algorithms,* Wiley.

Avruch, L. M., et al. (1995). A Spectroscopic Search for Protoclusters at High Redshift. *Bulletin of the American Astron. Society* 27(4).

Burke, G. J., and A. J. Poggio (1981). *Numerical Electromagnetics Code (NEC)—Method of Moments.* Rep. UCID18834, Lawrence Livermore Laboratory.

Linden, D. S. (1997a). *Automated Design and Optimization of Wire Antennas Using Genetic Algorithms.* Ph.D. dissertation, MIT.

Linden, D. S. (1997b). Using a Real Chromosome in a Genetic Algorithm for Wire Antenna Optimization. In *Proceedings of the IEEE APS International Symposium.*

Linden, D. S. (2000). Wire Antennas Optimized in the Presence of Satellite Structures Using Genetic Algorithms. In the *Proceedings of the IEEE Aerospace Conference.*

Linden, D. S., and E. E. Altshuler (1996). Automating Wire Antenna Design Using Genetic Algorithms. *Microwave Journal* 39(3).

Rahmat-Samii, Y., and E. Michielssen (eds.) (1999). *Electromagnetic Optimization by Genetic Algorithms.* Wiley.

Sandlin, B. S. (1997). *A Wire Antenna Designed for Space Wave Radiation Over the Earth Using a Genetic Algorithm.* Ph.D. dissertation, AFIT.

Weiss, P. (1999). On the Origin of Circuits. *Science News* 156:156–158.

21 CHAPTER

Evolutionary Techniques in Physical Robotics

Jordan B. Pollack Brandeis University
Hod Lipson Brandeis University
Sevan Ficici Brandeis University
Pablo Funes Brandeis University
Greg Hornby Brandeis University
Richard A. Watson Brandeis University

21.1 INTRODUCTION

Evolutionary and coevolutionary techniques have become a popular area of research for those interested in automated design. One of the cutting edge issues in this field is the ability to apply these techniques to real physical systems with all the complexities and affordances that such systems present. Here we present a selection of our projects, each of which advances the richness of the evolutionary substrate in one or more dimensions. We overview research in four areas: (1) high-parts-count static structures that are buildable, (2) the use of commercial CAD/CAM systems as a simulated substrate, (3) dynamic electromechanical systems with complex morphology that can be built automatically, and (4) evolutionary techniques distributed in a physical population of robots.

The field of robotics today faces a practical problem: Most problems in the physical world are too difficult for the current state of the art. The difficulties associated with designing, building, and controlling robots have led to a stasis (Moravec 1999), and robots in industry are only applied to simple and highly repetitive manufacturing tasks.

Even though sophisticated teleoperated machines with sensors and actuators have found important applications (exploration of inaccessible environments for example), they leave very little decision, if at all, to the on-board software (Morrison and Nguyen 1996). Hopes for autonomous robotics were raised high several times in the past—the early days of AI brought us Shakey the robot (Nilsson 1969), and the 1980s the "subsumption architecture" (Brooks 1991). Yet today, fully autonomous robots with minds of their own are not a reality.

The central issue we begin to address is how to get a higher level of complex physicality under control with less human design cost. We seek more controlled and moving mechanical parts, more sensors, more nonlinear interacting degrees of freedom—without entailing huge costs of human design, programming, manufacture, and operation. We suggest that this can be achieved only when robot design and construction are fully automatic and the constructions inexpensive enough to be disposable and/or recyclable.

The focus of our research is thus how to automate the integrated design of bodies and brains using a coevolutionary learning approach. Brain-body coevolution is a popular idea, but evolution of robot bodies is usually restricted to adjusting a few morphological parameters in an otherwise fixed, human-engineered automaton (Cliff and Noble 1997; Lee, Hallam, and Lund 1996; Lund, Hallam, and Lee 1997). We propose that the key is to evolve both the brain and the body, simultaneously and continuously, from a simple controllable mechanism to one of sufficient complexity for a task. We then require a replication process, that brings an exact copy of the evolved machine into reality. Finally, the transfer between a simulated environment and reality—the "reality gap"'—needs further embodied adaptation.

We see three technologies that are maturing past the threshold to make this possible. One is the increasing fidelity of advanced mechanical design simulation, stimulated by profits from successful software competition (Sincell 1999). The second is rapid, one-off prototyping and manufacture, which is proceeding from 3D plastic layering to stronger composite and metal (sintering) technology (Dimos, Danforth, and Cima 1999). The third is our understanding of coevolutionary machine learning in design and intelligent control of complex systems (Pollack and Blair 1998; Juillé and Pollack 1996; Angeline, Saunders, and Pollack 1994).

21.2 COEVOLUTION

Coevolution, when successful, dynamically creates a series of learning environments, each slightly more complex than the last, and a series of learners that are tuned to adapt in those environments. Sims's work (Sims 1994) on body-brain coevolution and the more recent Framsticks simulator (Komosinski and Ulatowski 1999) demonstrated that the neural controllers and simulated bodies could be coevolved. The goal of our research in coevolutionary robotics is to replicate and extend results from virtual simulations like these to the reality of computer-designed and -constructed special-purpose machines that can adapt to real environments.

We are working on coevolutionary algorithms to develop control programs operating realistic physical device simulators, both commercial off-the-shelf and our own custom simulators, where we finish the evolution inside real embodied robots. We are ultimately interested in mechanical structures that have complex physicality of more degrees of freedom than anything that has ever been controlled by human-designed algorithms, with lower engineering costs than currently possible because of minimal human design involvement in the product.

It is not feasible that controllers for complete structures could be evolved (in simulation or otherwise) without first evolving controllers for simpler constructions. Compared to the traditional form of evolutionary robotics (Floreano and Mondada 1996; Cliff, Harvey, and Husbands 1996; Lund 1995; Gallagher et al. 1996; Kawauchi, Inaba, and Fukuda 1995), which serially downloads controllers into a piece of hardware, it is relatively easy to explore the space of body constructions in simulation. Realistic simulation is also crucial for providing a rich and nonlinear universe. However, while simulation creates the ability to explore the space of constructions far faster than real-world building and evaluation could, transfer to real constructions is problematic. Because of the complex emergent interactions between a machine and its environment, learning and readaptation must occur in "embodied" form as well (Jakobi 1997; Mataric and Cliff 1996; Floreano 1998).

21.3 RESEARCH THRUSTS

We have four major thrusts in achieving fully automated design (FAD) and manufacture of high-parts-count autonomous robots. The first is evolution inside simulation, but in simulations more and more realistic so the results are not simply visually believable, as in Sims's work, but also tie into manufacturing processes. Indeed, interfacing evolutionary computation systems to commercial off-the-shelf CAD/CAM systems through developer interfaces to mechanical simulation programs seems as restrictive as developing programming languages for 8K memory microcomputers in the mid-1970s. However, even though the current mechanical simulation packages are "advisory"' rather than blueprint generating, and are less efficient than research code, as computer power grows and computer-integrated manufacturing expands, these highly capitalized software products will absorb and surpass research code, and moreover will stay current with the emerging interfaces to future digital factories. The second thrust is to evolve automatically buildable machines, using custom simulation programs. Here, we are willing to reduce the universe of mechanisms we are working with in order to increase the fidelity and efficiency of the simulators and reduce the cost of building resulting machines. The third is to perform evolution directly inside real hardware, which escapes the known limitations of simulation and defines a technology supporting the final learning in embodied form. This is perhaps the hardest task because of the power, communication, and other reality constraints. The fourth thrust addresses handling high-parts-count structures with realistic complexity. We have preliminary and promising results in each of these four areas, which we outline below.

21.4 EVOLUTION IN SIMULATION

We have been doing evolution of neural-network controllers inside realistic CAD simulations as a prelude to doing body reconfiguration and coevolution. Our lab is using a CAD/CAM software package that comprises a feature-based solid-modeling system.[1] Widely used in industry, it includes a mechanical simulation component that can simulate the function of real-world mechanisms, including gears, latches, cams, and stops. This program has a fully articulated development interface to the C programming language, which we have used in order to interface its models to our evolutionary recurrent neural network software.

To date, we have used this system with evolved recurrent neural controllers for one- and two-segment inverted pendulums and for Luxo (an animated lamp creature, Figure 21.1). Many researchers have evolved such controllers in simulation, but no one has continuously deformed the simulation and brought the evolved controllers along, and no one else has achieved neural control inside commercial simulations. We believe this should lead to easy replication, extension, and transfer of our work.

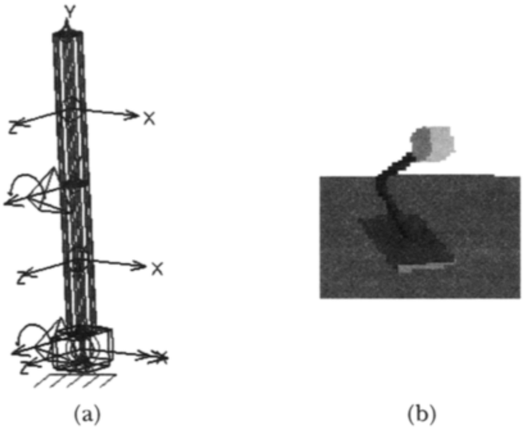
We have successful initial experiments consisting of evolving recurrent neural network controllers for the double-pole balancing problem, where we slowly "morphed"' the body simulator by simulating a stiff spring at the joint connecting the two poles and relaxing its stiffness.

Some of the ways to achieve continuous body deformation are the following:

- New links can be introduced with "no-op" control elements.
- The mass of new links can initially be very small and then incremented.
- The range of a joint can be small and then given greater freedom.

---

[1] Parametric Technology Corporation's Pro/Engineer.



(a) (b)

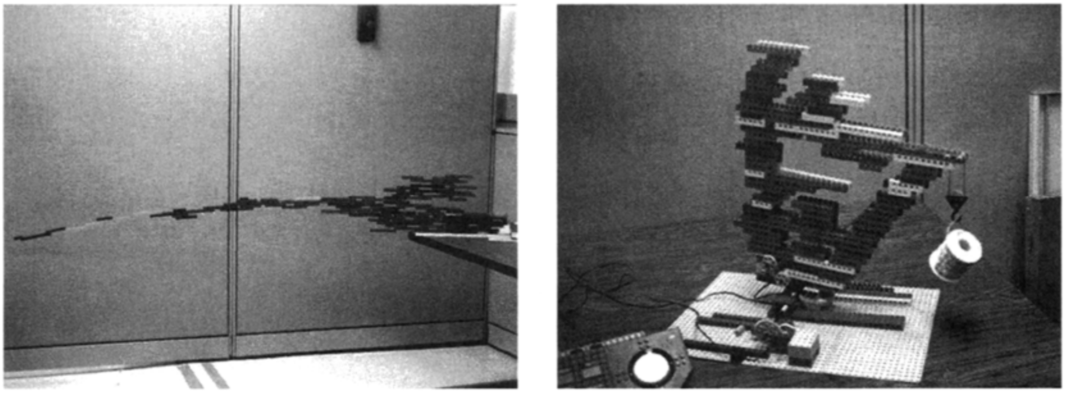
21.1 FIGURE Commercial CAD models for which we evolved recurrent neural net controllers: (a) a two-segment inverted pendulum; (b) a Luxo lamp.

- A spring can be simulated at a joint and the spring constant relaxed.
- Gravity and other external load forces can be simulated lightly and then increased.

21.5 BUILDABLE SIMULATION

Commercial CAD models are in fact not constrained enough to be buildable because they assume a human provides numerous constraints to describe reality. In order to evolve both the morphology and behavior of autonomous mechanical devices that can be built, one must have a simulator that operates under many constraints, and a resultant controller that is adaptive enough to cover the gap between the simulated and real world. Features of a simulator for evolving morphology are the following:

- *Representation:* It should cover a universal space of mechanisms.
- *Conservative:* Because simulation is never perfect, it should preserve a margin of safety.
- *Efficient:* It should be quicker to test in simulation than through physical production and test.
- *Buildable:* Results should be convertible from a simulation to a real object.



(a) (b)

21.2 FIGURE (a) FAD LEGO bridge (cantilever) and (b) crane (triangle). © Pablo Funes and Jordan Pollack, used by permission.

One approach is to custom-build a simulator for modular robotic components and then evolve either centralized or distributed controllers for them. In advance of a modular simulator with dynamics, we recently built a simulator for (static) LEGO bricks and used very simple evolutionary algorithms to create complex LEGO structures, which were then manually constructed (Funes and Pollack 1997, 1998, 1999).

Our model considers the union between two bricks as a rigid joint between the centers of mass of each one, located at the center of the actual area of contact between them. This joint has a measurable torque capacity. That is, more than a certain amount of force applied at a certain distance from the joint will break the two bricks apart. The fundamental assumption of our model is this idealization of the union of two LEGO bricks.

The genetic algorithm reliably builds structures that meet simple fitness goals, exploiting physical properties implicit in the simulation. Building the results of the evolutionary simulation (by hand) demonstrated the power and possibility of fully automated design. The long bridge of Figure 21.2(a) shows that our simple system discovered the cantilever, while the weight-carrying crane of Figure 21.2(b) shows it discovered the basic triangular support.

21.6 EVOLUTION AND CONSTRUCTION OF ELECTROMECHANICAL SYSTEMS

The next step is to add dynamics to modular buildable physical components. We are experimenting with a new process in which both robot morphology and control evolve in simulation and then replicate automatically into reality. The robots are comprised of only linear actuators and sigmoidal control neurons embodied in an arbitrary thermoplastic body. The entire configuration is evolved for a particular task, and selected individuals are printed preassembled (except motors) using 3D solid printing (rapid prototyping) technology, later to be recycled into different forms. In doing so, we establish for the first time a complete physical evolution cycle. In this project, the evolutionary design approach assumes two main principles: (1) to minimize inductive bias, we must strive to use the lowest-level building blocks possible, and (2) we coevolve the body and the control, so that that they stimulate and constrain each other.

We use arbitrary networks of linear actuators and bars for the morphology, and arbitrary networks of sigmoidal neurons for the control. Evolution is simulated starting with a soup of disconnected elements and continues over hundreds of generations of hundreds of machines, until creatures that are sufficiently proficient at the given task emerge. The simulator used in this research is based on quasi-static motion. The basic principle is that motion is broken down into a series of statically stable frames solved independently. While quasi-static motion cannot describe high-momentum behavior such as jumping, it can accurately and rapidly simulate low-momentum motion. This kind of motion is sufficiently rich for the purpose of the experiment and, moreover, it is simple to induce in reality since all real-time control issues are eliminated (Lipson and Pollack 2000a, 2000b).

Several evolution runs were carried out for the task of locomotion. Fitness was awarded to machines according to the absolute average distance traveled over a specified period of neural activation. The evolved robots exhibited various methods of locomotion, including crawling, ratcheting, and some forms of pedalism (see Color Plate 18).

Selected robots are then replicated into reality: Their bodies are first fleshed to accommodate motors and joints, and then copied into material using rapid prototyping technology. Temperature-controlled print heads extrude thermoplastic material layer by layer, so that the arbitrarily evolved morphology emerges preassembled as a solid three-dimensional structure without tooling or human intervention. Motors are then snapped in, and the evolved neural network is activated (Color Plate 19). The robots then perform in reality as they did in simulation.

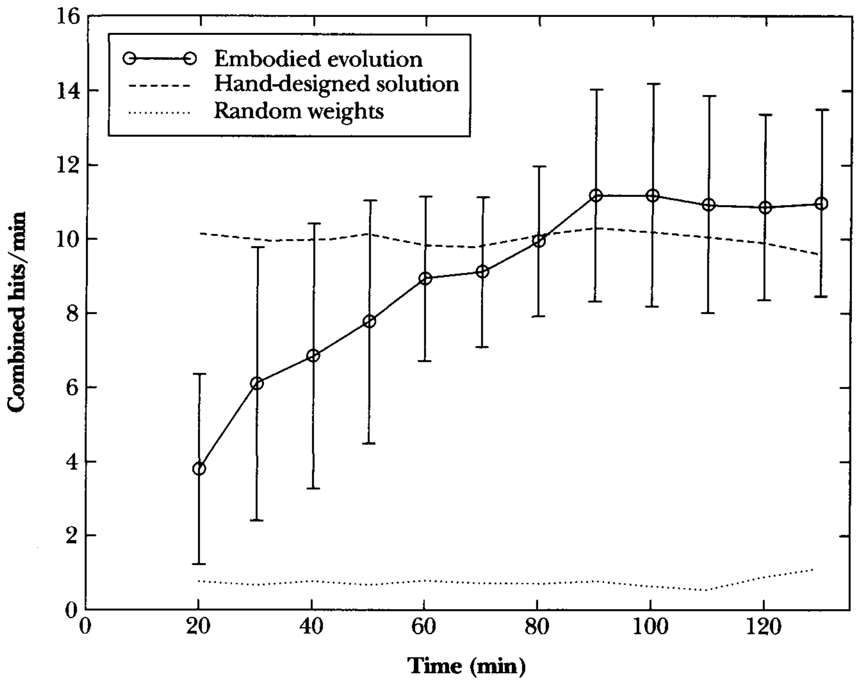
21.7 EMBODIED EVOLUTION

Once a robot is built, it may well be necessary to "fine-tune'" adaptation in the real world. Our approach is to perform adaptation via a decentralized evolutionary algorithm that is distributed and embodied within a population of robots. The distributed and asynchronous operation of this evolutionary method allows the potential for being scaled to very large populations of robots, on the order of hundreds or thousands, thus enabling speedup that is critical when using evolution in real robots.

Technologically, this introduces two main problems: long-term power and reprogramming (Watson, Ficici, and Pollack 1999; Ficici, Watson, and Pollack 1999). Many robots' batteries last only for a few hours, and robots typically have to be attached to a PC for new programs to be uploaded. In order to do large group robot learning experiments, we have designed a continuous-power floor system, and utilized infrared (IR) communications to transfer programs between robots. We are thus able to run a population of learning robots battery-free and wire-free for days at a time (Color Plate 19). Evolution is not directed by a central off-board computer that installs new programs to try out, but rather is distributed into the behavior of all the robots. The robots exchange program specifications with each other, and this "culture" is used to "reproduce" the more successful behaviors and achievement of local goals.

The control architecture is a simple neural network and the specifications for it are evolved online. Each robot tries parameters for the network and evaluates its own success. The more successful a robot is at the task, the more frequently it will broadcast its network specifications via its local-range IR communications channel. If another robot happens to be in range of the broadcast, it will adopt the broadcast value with a probability inversely related to its own success rate. Thus, successful robots attempt to influence others, and resist the influence of others, more frequently than less successful robots. We have shown this paradigm to be robust both in simulation and in real robots, allowing for the parallel, asynchronous evolution of large populations of robots with automatically developed controllers. These controllers compare favorably to human designs and often surpass them when human designs fail to take all-important environmental factors into account. Figure 21.3 shows averaged runs of the robots in a light-seeking task, comparing evolved controllers to random and human-designed ones.

Our research goals in this area involve group interactive tasks based on multiagent systems, such as group pursuit and evasion, box pushing, and team games. These domains have been out of reach of traditional evolutionary



21.3 FIGURE Averaged runs of robots in a "light-gathering" task, with various controllers.

methods and typically are approached with hand-built (nonlearning) controller architectures (Beckers, Holland, Deneubourg 1994; Balch and Arkin 1995; Rus, Donald, and Jennings 1995; Donald, Jennings, and Rus 1997). Work that does involve learning typically occurs in simulation (Tan 1993; Littman 1994; Saunders and Pollack 1996; Balch 1997), or in relatively simple physical domains (Mahadevan and Connell 1991; Mataric 1994a, 1994b; Parker 1997; Uchibe, Asada, and Hosoda 1998).

21.8 CONCLUSIONS

Can evolutionary and coevolutionary techniques be applied to real physical systems? In this chapter we have presented a selection of our work, each of which addresses physical evolutionary substrate in one or more dimensions. We have overviewed research in handling high-parts-count static structures that are buildable, use of commercial CAD/CAM systems as a realistic simulated substrate, dynamic electromechanical systems with complex morphology that can be built automatically, and evolutionary techniques distributed in a physical population of robots.

Our long-term vision is that both the morphology and control programs for self-assembling robots arise directly through hardware and software coevolution. Primitive active structures that crawl over each other, attach and detach, and accept temporary employment as supportive elements in "corporate" beings can accomplish a variety of tasks, if enough design intelligence is captured to allow true self-configuration rather than human redeployment and reprogramming. When tasks cannot be solved with current parts, new elements are created through fully automatic design and rapid prototype manufacturing.

Our current research moves toward the overall goal down multiple interacting paths, where what we learn in one thrust aids the others. We envision the improvement of our hardware-based evolution structures, expanding focus from static buildable structures and unconnected groups to reconfigurable active systems governed by a central controller, and then the subsequent parallelization of the control concepts. We see a path from evolution inside CAD/CAM and buildable simulation, to rapid automatic construction of novel controlled mechanisms, from control in simulation to control in real systems, and finally from embodied evolution of individuals to the evolution of heterogeneous groups that learn by working together symbiotically. We believe such a broad program is the best way to ultimately construct complex autonomous robots that are self-organizing and self-configuring corporate assemblages of simpler automatically manufactured parts.

ACKNOWLEDGMENTS

This research was supported in part by the National Science Foundation (NSF), the office of Naval Research (ONR), and the Defense Advanced Research Projects Agency (DARPA). Thanks to the MIT Media Lab for supplying the microcontrollers used in our embodied evolution experiments.

REFERENCES

Angeline, P. J., G. M. Saunders, and J. B. Pollack (1994). An Evolutionary Algorithm That Constructs Recurrent Networks. *IEEE Transactions on Neural Networks* 5:54–65.

Balch, T. (1997). Learning Roles: Behavioral Diversity in Robot Teams. In *1997 AAAI Workshop on Multiagent Learning,* AAAI Press.

Balch, T., and R. Arkin (1995). Motor Schema-Based Formation Control for Multiagent Robot Teams. In *Proceedings of the First International Conference on Multiagent Systems* ICMAS-95, AAAI Press, pp. 10–16.

Beckers, R., O. Holland, and J. Deneubourg (1994). From Local Actions to Global Tasks: Stigmergy and Collective Robotics. In R. Brooks and P. Maes (eds.), *Artificial Life IV,* MIT Press, pp. 181–189.

Brooks, R. (1991). Intelligence without Representation. *Artificial Intelligence* 47:139–160.

Cliff, D., I. Harvey, and P. Husbands (1996). Evolution of Visual Control Systems for Robot. In M. Srinivisan and S. Venkatesh (eds.), *From Living Eyes to Seeing Machines,* Oxford.

Cliff, D., and J. Noble (1997). Knowledge-based Vision and Simple Visual Machines. *Philosophical Transactions of the Royal Society of London,* Series B, 352:1165–1175.

Dimos, D., S. Danforth, and M. Cima (1999). Solid Freeform and Additive Fabrication. In J. A. Floro (ed.), *Growth Instabilities and Decomposition during Heteroepitaxy,* Elsevier.

Donald, B., J. Jennings, and D. Rus (1997). Minimalism + Distribution = Supermodularity. *Journal on Experimental and Theoretical Artificial Intelligence* 9:293–321.

Ficici, S. G., R. A. Watson, and J. B. Pollack (1999). Embodied Evolution: A Response to Challenges in Evolutionary Robotics. In J. L. Wyatt and J. Demiris, (eds.), *Eighth European Workshop on Learning Robots,* pp. 14–22.

Floreano, D. (1998). Evolutionary Robotics in Artificial Life and Behavior Engineering. In T. Gomi (ed.), *Evolutionary Robotics,* AAI Books.

Floreano, D., and F. Mondada (1996). Evolution of Homing Navigation in a Real Mobile Robot. *IEEE Transactions on Systems, Man, and Cybernetics.*

Funes, P., and J. B. Pollack (1997). Computer Evolution of Buildable Objects. In P. Husbands, and I. Harvey (eds.), *Fourth European Conference on Artificial Life,* MIT Press, pp. 358–367.

Funes, P., and J. B. Pollack (1998). Evolutionary Body Building: Adaptive Physical Designs for Robots. *Artificial Life* 4:337–357.

Funes, P., and J. B. Pollack (1999). Computer Evolution of Buildable Objects. In P. Bentley (ed.), *Evolutionary Design by Computers,* Morgan Kaufmann, pp. 387–403.

Gallagher, J. C., R. D. Beer, K. S. Espenschield, and R. D. Quinn (1996). Application of Evolved Locomotion Controllers to a Hexapod Robot. *Robotics and Autonomous Systems* 19:95–103.

Jakobi, N. (1997). Evolutionary Robotics and the Radical Envelope of Noise Hypothesis. *Adaptive Behavior* 6:131–174.

Juillé, H., and J. B. Pollack (1996). Dynamics of Co-evolutionary Learning. In *Proceedings of the Fourth International Conference on Simulation of Adaptive Behavior,* MIT Press, pp. 526–534.

Kawauchi, Y., M. Inaba, and T. Fukuda (1995). Genetic Evolution and Self-organization of Cellular Robotic System. *JSME Int. J. Series C. (Dynamics, Control, Robotics, Design & Manufacturing)* 38:501–509.

Komosinski, M., and S. Ulatowski (1999). Framsticks: Towards a Simulation of a Nature-like World, Creatures and Evolution. In J. D. N. Dario Floreano and F. Mondada (eds.), *Proceedings of 5th European Conference on Artificial Life* (ECAL99), Lecture Notes in Artificial Intelligence 1674, Springer-Verlag, pp. 261–265.

Lee, W., J. Hallam, and H. Lund (1996). A Hybrid GP/GA Approach for Co-evolving Controllers and Robot Bodies to Achieve Fitness-Specified Tasks. In *Proceedings of IEEE 3rd International Conference on Evolutionary Computation,* IEEE Press, pp. 384–389.

Lipson, H., and J. B. Pollack (2000a). Evolution of Machines. In *Proceedings of 6th International Conference on Artificial Intelligence in Design* (to appear).

Lipson, H., and J. B. Pollack (2000b). Towards Continuously Reconfigurable Robotics. In *Proceedings of IEEE International Conference on Robotics and Automation* (to appear).

Littman, M. (1994). Markov Games as a Framework for Multi-agent Reinforcement Learning. In *Proceedings of the International Machine Learning Conference,* pp. 157–163.

Lund, H. (1995). Evolving Robot Control Systems. In I. Alexander (ed.), *Proceedings of 1NWGA,* University of Vaasa.

Lund, H., J. Hallam, and W. Lee (1997). Evolving Robot Morphology. In *Proceedings of IEEE Fourth International Conference on Evolutionary Computation,* IEEE Press, pp. 197–202.

Mahadevan, S., and H. Connell (1991). Automatic Programming of Behavior-based Robots Using Reinforcement Learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence* (AAAI '91), pp. 8–14.

Mataric, M. (1994a). Learning to Behave Socially. In D. Cliff, P. Husbands, J. A. Meyer, and S. Wilson (eds.), *From Animals to Animats 3,* MIT Press, pp. 453–462.

Mataric, M. (1994b). Reward Functions for Accelerated Learning. In W. Cohen, and H. Hirsh (eds.), *Proceedings of the Eleventh International Conference on Machine Learning,* Morgan Kaufmann, pp. 181–189.

Mataric, M. J., and D. Cliff (1996). Challenges in Evolving Controllers for Physical Robots. *Robotics and Autonomous Systems* 19:67–83.

Moravec, H. P. (1999). Rise of the Robots. *Scientific American:* 124–135.

Morrison, J., and T. Nguyen (1996). On-board Software for the Mars Pathfinder Microrover. In *Proceedings of the Second IAA International Conference on Low-Cost Planetary Missions,* John Hopkins University Applied Physics Laboratory.

Nilsson, N. J. (1969). A Mobile Automaton: An Application of Artificial Intelligence Techniques. In *Proceedings of the International Joint Conference on Artificial Intelligence,* pp. 509–520.

Parker, L. (1997). Task-oriented Multi-robot Learning in Behavior-based Systems. *Advanced Robotics, Special Issue on Selected Papers from IROS '96* 11:305–322.

Pollack, J. B., and A. D. Blair (1998). Coevolution in the Successful Learning of Backgammon Strategy. *Machine Learning* 32:225–240.

Rus, D., B. Donald, and J. Jennings (1995). Moving Furniture with Teams of Autonomous Robots. In *Proceedings of IEEE/RSJ IROS'95*, pp. 235–242.

Saunders, G., and J. Pollack (1996). The Evolution of Communication Schemes of Continuous Channels. In P. Maes, M. Mataric, J. A. Meyer, J. Pollack, and S. Wilson (eds.), *From Animals to Animats IV,* MIT Press, pp. 580–589.

Sims, K. (1994). Evolving 3D Morphology and Behavior by Competition. In R. Brooks and P. Maes (eds.), *Proceedings 4th Artificial Life Conference,* MIT Press.

Sincell, M. (1999). Physics Meets the Hideous Bog Beast. *Science* 286:398–399.

Tan, M. (1993). Multi-agent Reinforcement Learning: Independent vs. Cooperative Agents. In *Proceedings of the Tenth International Machine Learning Conference,* pp. 330–337.

Watson, R., S. Ficici, and J. Pollack (1999). Embodied Evolution: Embodying an Evolutionary Algorithm in a Population of Robots. In P. Angeline, Z. Michalewicz, M. Schoenauer, X. Yao, and A. Zalzala (eds.), *1999 Congress on Evolutionary Computation.*

Uchibe, E., M. Asada, and K. Hosoda (1998). Cooperative Behavior Acquisition in Multi Mobile Robots Environment by Reinforcement Learning Based on State Vector Estimation. In *Proceedings of International Conference on Robotics and Automation,* pp. 1558–1563.

This Page Intentionally Left Blank

22 CHAPTER Patenting Evolved Bactericidal Peptides

Shail Patel Unilever Research
Ian Stott Unilever Research
Manmohan Bhakoo Unilever Research
Peter Elliott Unilever Research

22.1 INTRODUCTION

Peptides are molecules formed from a string of amino acids. There are 20 naturally occurring amino acids, usually denoted by letters of the alphabet, and any particular peptide is uniquely identified by the particular sequence of amino acids. Peptides are of great interest for industrial, pharmaceutical, and medical purposes, as they are quintessentially natural, and therefore find acceptance with the public in many applications. This chapter is concerned with the evolution of new bactericidal peptides, that is, peptides that kill bacteria for a wide range of common applications such as kitchen cleaners, food preservatives, and so on.

Significant advances have been made in the use of computer-aided molecular design (CAMD) techniques for the design of novel molecules possessing desired properties. For example, polymers have been designed that have favorable properties in terms of glass transition, resistivity, and conductivity (Venkatasubramaniam, Chan, and Caruthers 1994; Maranas 1997) or in mechanical and shrinkage (de Weijer et al. 1993) properties. New proteins have been designed with folding (Jones 1994; Hellinga and Richards 1994; Jin, Leungn, and Weaver 1997), binding (Stadler 1997), or cleavage (Schneider, Schuchhardt, and Wrede 1995b) properties. There are a number of good reviews of this general field (Clark and Westhead 1996; Mavrovounitis 1996; Parril 1997).

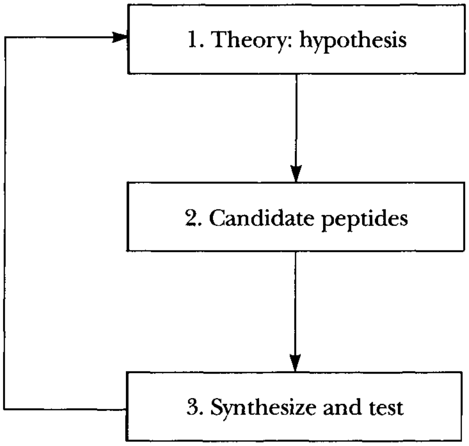
The computer-aided molecular design approach is best used in parallel with more conventional "synthesize-and-test" methods, and typically involves two stages:

1. *Forward modeling:* This refers to the use of linear or nonlinear modeling methods such as partial least squares (PLS) or neural networks (NN) to predict molecular properties, often called quantitative structure activity relationships (QSARs). These methods build predictive models based on experimental data. They may use molecular parameters derived from the 3D structure of the peptide, or a structural description of the molecule (Venkatasubramanian, Chan, and Caruthers 1994, Venkatasubramanian et al. 1994).
2. *Model inversion/optimization to discover new molecules:* For linear models, it is possible to simply invert the equations. For nonlinear models, the inversion is generally one-to-many, that is, there are many molecules that have the same activity, and so the inversion is more reasonably treated as an optimization problem in the space of all molecules of the particular class. Genetic algorithms have been found to be a successful way of designing molecules in this way (Jones 1994; Venkatasubramanian, Chan, and Caruthers 1995; Schneider, Schuchhardt, and Wrede 1995b; Donguet, Thoreau, and Grassy 2000; Schneider et al. 2000).

This chapter reports on a successful application of evolutionary peptide design (Patel et al. 1998). New peptides have been evolved by a genetic algorithm using a neural net QSAR as an objective function (i.e., searching for peptides that the neural net model predicts to be bactericidally active). A selection of these peptides has then been synthesized, experimentally tested, and found to be active. These peptides may have commercial value, and so it is necessary to protect the intellectual property rights associated with them. It would be impractical to list all the peptides that the genetic algorithm evolves, so in an extension of established patenting practice, we framed a patent application that used the neural net QSAR model to define the regions of peptide space that we claimed within the patent (Bhakoo, Patel, and Stott 1996). The neural net predictive QSAR model is not itself patented; it is fully disclosed, as are the methods of deriving the molecular parameters, so that anyone "skilled in the art" may reproduce the results and models we have obtained. What the patent seeks to protect is the particular set of peptides obtained by application of our CAMD techniques.

22.2 DESIGN CYCLE

A conventional "synthesize-and-test" method starts off with a hypothesis for the mechanism of action, generates a number of candidate molecules to test, conducts experiments on a selection of these, and uses the results to confirm or



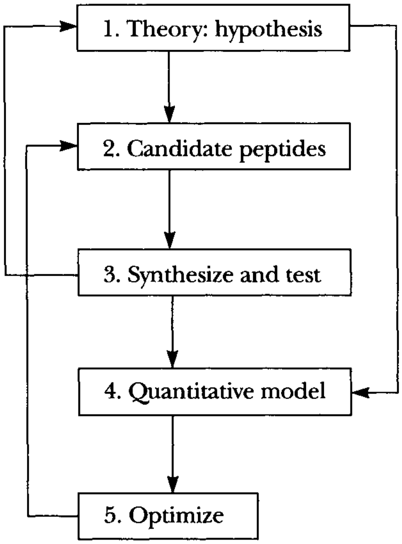
22.1 FIGURE

Conventional synthesize-and-test methodology draws on (1) a theoretically based hypothesis of the mechanism of action to (2) generate a set of peptides that may prove or disprove the hypothesis. (3) A selection of these is synthesized and tested for activity.

falsify the hypothesis. Figure 22.1 shows this primary loop. A CAMD approach puts in place a secondary loop of modeling and optimization where the two steps of forward modeling and model inversion form part of a total cycle of activity (Figure 22.2), which consists of the following:

1. Develop/refine theory-based hypotheses for the mechanism(s) of action.
2. Generate candidate peptides based on these hypotheses.
3. Perform experiments, including synthesizing and testing a number of candidate molecules.
4. Develop a quantitative model, including building an empirical model relating selected descriptive parameters for the molecule to its activity.
5. Optimize/search in the space of peptides using the quantitative model as an objective function.

Once the primary loop has generated a sufficient quantity of data for modeling, the secondary modeling loop may commence, generating candidate peptides based on the experimental data. Note that the models and the results of the optimization may also be used to influence the refinement of the theoretically based hypotheses. This process is closely aligned to the hypothetico-deductive scientific process, and makes a compelling argument for placing computer-aided design techniques firmly alongside traditional synthesize-and-test methods.

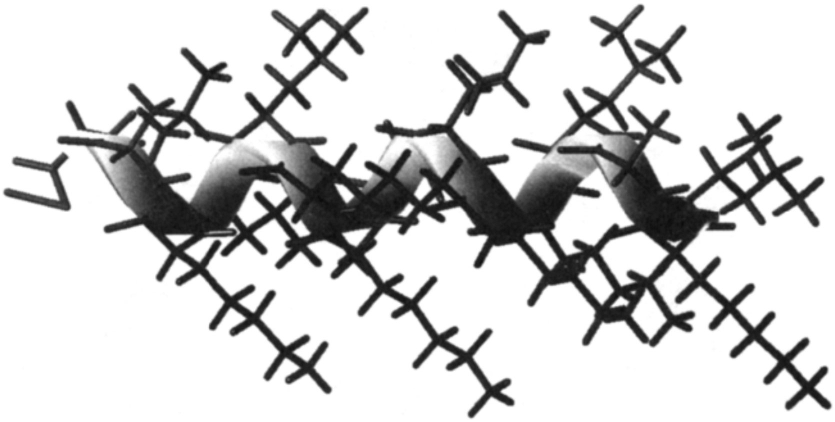


22.2 FIGURE Computer-aided molecular design builds on the synthesize-and-test method. Descriptive parameters based on the mechanism of action are selected or created together with data generated by the experimental synthesize-and-test loop.

22.3 HYPOTHESIS: MECHANISM OF ACTION

We are concerned with the design of novel peptides with a broad spectrum of bactericidal activity. Such peptides should be capable of crossing through a cell wall or outer cell wall membrane and then disrupting or disintegrating cell membranes. This is particularly relevant to the cytoplasmic membrane, which is a selective barrier that restricts entry and exit of solutes. This results in irreversible osmotic-colloidal interactions that kill the bacteria by dramatically altering the proton motive force, or potential across the membrane. The peptides suitable for this purpose should be capable of forming ion channels in membranes by aggregation (pseudoionophores) and insertion in order to span the membranes (Wade et al. 1992; Christensen 1998; Hancock and Poxton 1988).

The minimum length to span a membrane of width around 2.5–4.0 nm is at least 15 amino acid residues (Hancock and Poxton 1988; Veld, Drissen, and Konnings 1993). The peptides should be alphahelical to be transmembrane—that is they should form a helix along the chain of amino acids (Figure 22.3). The peptide should also be amphiphilic and laterally amphipathic, with one face of the helix displaying hydrophobic amino acids and the opposite face displaying



22.3
FIGURE

Side view of an amphiphilic and laterally amphipathic peptide, with one face of the helix displaying hydrophobic residues, and the opposite face displaying hydrophilic residues.

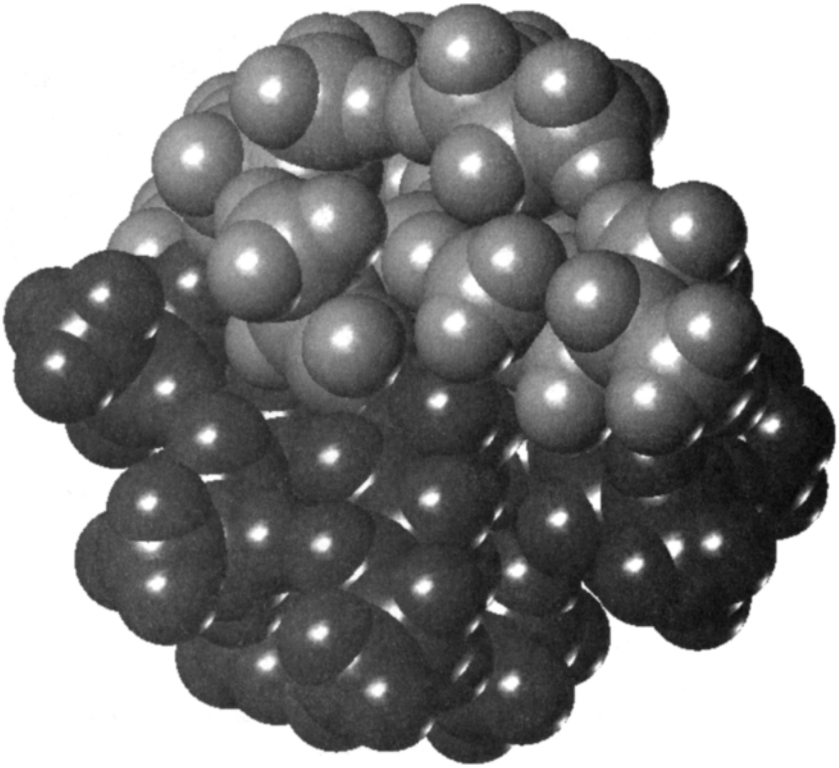
hydrophilic amino acids (Figure 22.4). A number of polar, acidic, or basic amino acids are required within the peptide to impart suitable solubility characteristics (Fasman 1989).

Transmembrane pores or channels may arise from the insertion of monomeric peptides, where the pore is part of the secondary structure of the monomer. Or they may arise from aggregated monomers, which form oligomeric peptides, resulting in a "barrel-stave" type of pore (Bhakoo, Birkbeck, and Fleer 1985). Figure 22.5 shows an end view of six peptides aggregated to form a pore, which forms an ion channel through the membrane.

These theories concerning the mechanism of action form a set of "design rules," which can then be used to design peptides that should be active. The CAMD approach implicitly captures such rules in a quantitative model that, given a particular "virtual" peptide, can predict its activity. We have pursued both routes in parallel.

22.4 EXPERIMENTAL MEASURES AND MODELING TECHNIQUES

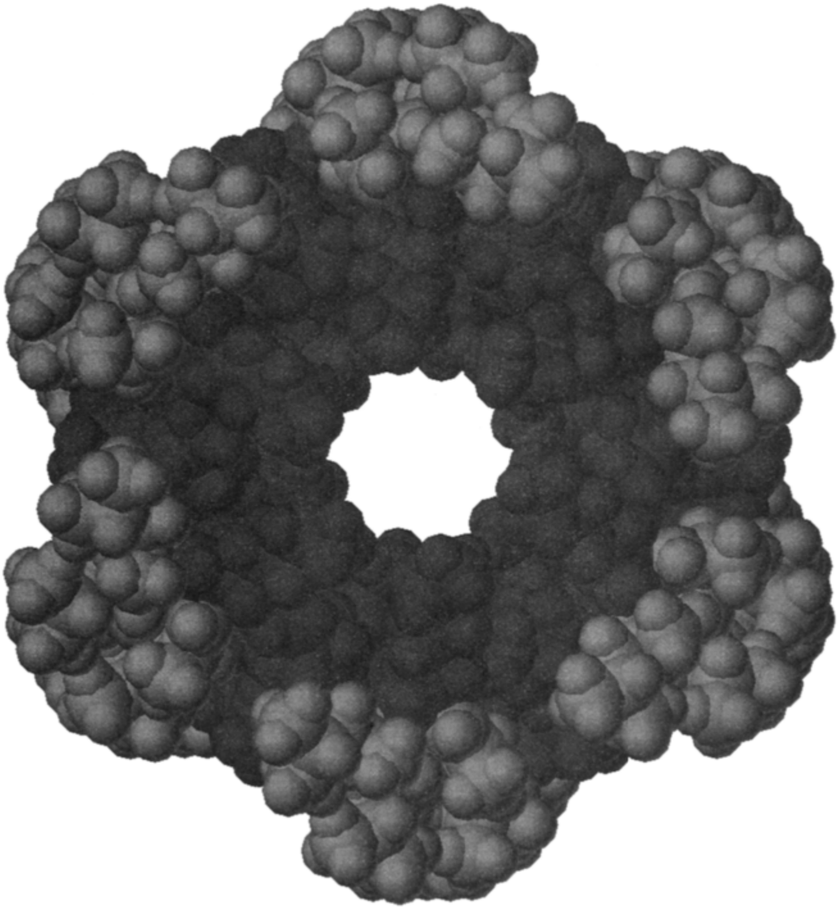
Test bacteria *Staphylococcus aureus* ATCC 6538 and *Escherichia coli* ATCC 11229 were cultured overnight, harvested, and suspended in Ringer's solution to obtain appropriate initial cell numbers (10^6–10^8). The bacteria were incubated with the designed test peptides (various concentrations) aseptically in 96 well



22.4 FIGURE Space-filling view of the peptide in Figure 22.3 looking down the helical axis.

microtitre plates and relevant controls were taken. The bactericidal activity (bacterial killing) was measured by a total viable counts (TVC) technique after an incubation period of 2 hours with the test peptides, and the results expressed as log bacterial kill. The log bacterial kill is calculated as the difference in log scale of bacteria initially present and the number of bacteria still viable after incubation with peptides.

Over the period of study, a total of 29 peptides were synthesized and tested for the bactericidal activity, and a further five were synthesized and tested based on the model predictions (see Table 22.1). We present here in detail the modeling of bactericidal activity against *S. aureus;* the case against *E. coli* runs in a similar vein. In this study, we adopt a two-stage approach to modeling: The first step is to generate a number of molecular descriptors mainly based on the 3D structure of the peptides, and the second step is to find a relationship between these parameters and the bactericidal activity (Figure 22.6).



22.5 FIGURE Space-filling view of an aggregation of six peptides showing a "barrel-stave" pore, which may form an ion channel in the cell membrane. The orientation follows that of Figure 22.4.

22.4.1 Molecular Modeling

Based on the mechanism of action, 39 molecular parameters were selected or created (using the TRIPOS-Sybyl software[1]) that might explain the bactericidal activity against *S. aureus* (see Table 22.2). Given an appropriate peptide length of greater than 15 amino acid residues, we assume

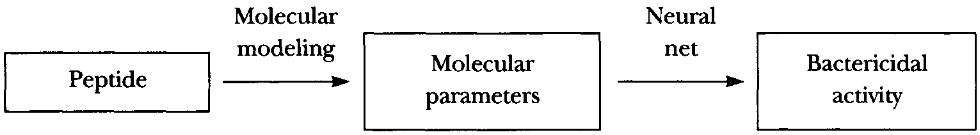
$$\text{bactericidal activity} = f\,(\text{diffusion, aggregation})$$

---

[1] Sybyl, TRIPOS, Inc., 1699 Hanley Rd., St. Louis, Missouri, 63144–2913.

		S. aureus		*E. coli*	
Example	**Sequence**	**Measured**	**Predicted**	**Measured**	**Predicted**
MB_03	VSSKYLSKVKVKRGK	4	4.18	1.63	0.75
MB_04	ARLAKKALRRLAKKD	1.46	1.08	0.83	0.44
MB_10	GESLASKAAKKAAER	0.78	0.92	0.46	0.45
MB_15	ESLAKALSKEALKALK	1	0.74	1.24	0.44
MB_18	**LKALKKLAKKLKKLA**	**7**	**6.70**	**4.17**	**6.28**
MB_22	GWLLLEYIPVIAAL	0.54	0.49	0.41	0.44
MB_31	EAALKAALDLAAKLA	0.75	1.13	0.39	0.45
MB_00	**LKLLKKLLKKLKKLL**	**6.94**	**6.90**	**7.15**	**6.02**
MB_21	**FASLLGKALKALAKQ**	**6**	**6.32**	**6.15**	**5.68**
MB_25	LSSALSALSSALSSK	0.54	0.46	0.37	0.44
MB_32	ERSAAKSAARSLARR	0.67	0.46	0.08	0.44
MB_33	EKTLARTAAKTALKK	0.43	0.58	0.22	0.44
MB_34	EKAAAKSAAAKTLARR	0.43	0.33	0.24	0.44
MB_35	VSSKYLSKALVKAGR	0.54	1.38	0.26	0.45
MB_36	**FASLLGKALKALLAKLAKQ**	**5.74**	**4.92**	**5.95**	**5.40**
MB_37	**FASLLGKLAKKLAKKALK**	**5.74**	**5.91**	**5.22**	**4.96**
MB_38	ESLKARSLKKSLKLKKLL	1.58	1.38	0.58	0.45
MB_41	ELAKKALKALKKALKSAR	3.64	3.33	0.18	0.59
MB_43	ELAKKALRALKKALKSAK	2.75	3.07	0.22	0.54
MB_45	ETFAKKALKALEKLLKKG	2.77	2.91	0.16	0.57
CM_1	**LALLKVLLRKIKKAL**	**5.68**	**5.85**	**4**	**3.54**
CM_2	LULLLKILLLKKLKA	3.38	3.14	0.75	0.83
CM_3	**ALKAALLAILKIVRVIKK**	**5.68**	**5.47**	3.07	4.07
CM_4	LLAILLLALLALRKKVLA	0.99	1.34	0.38	0.46
MB_40	ETELAKKALKALKLKKLA	0.47	0.32	0.16	0.44
MB_46	ESSLKKKALSKLSKLLKKG	2.57	2.63	0.28	0.51
MB_47	**QKAASRLLRALSKLLEAF**	**5.65**	**5.55**	0.11	0.48
MB_48	QKALAKLAKKALKALAKQ	1.03	1.31	1.77	0.45
MB_50	ESKAAKAAKKAAKAKASE	0.24	0.40	0.18	0.44
MC_03	**AASKAAKTLAKLLSSLLKL**	**5.96**	**7.24**	1.96	1.72
MC_04	**LLKKLLRAASKALSLL**	**5.9**	**7.13**	0.45	0.82
MC_05	**AAKKLSKLLKTLLKLL**	**5.76**	**7.36**	1.01	2.06
MC_08	**KALKKLLKLASSLLTAL**	**5.9**	**7.06**	1.61	1.12
MC_10	**AASKALRTASRLARSLLT**	**5.85**	**7.03**	0.41	0.56

22.1 TABLE Primary peptide sequences with measured and predicted activity.



22.6 FIGURE The first step of molecular modeling generates a number of molecular parameters based on the general characteristics and the 3D structure of the peptide. These parameters are used as input to a neural network to give a prediction of the bactericidal activity.

Molecular Parameter Class	**Number of Parameters**	**Molecular Parameters**
Simple	4	Molecular weight, length of the alpha helix, cross-sectional area of the alpha helix, and length/weight
Hydrophobic dipole	5	x, y, z, and zy components and magnitude
Hydrophobicity	3	Sum of the hydrophobicity of the hydrophobic amino acids, sum of the hydrophobicity of the hydrophilic amino acids, and sum of these two parts
Solvent accessible surface area	3	Hydrophobic, hydrophilic and hydrophilic/hydrophobic
Charge dipole	4	x, y, and z components and magnitude
Charge	4	Sum of the charge of positively charged amino acids, sum of the charge of positively charged amino acids, sum and difference of these two parts
Fit of hydrophobicity of amino acids to different patterns	2	Fit to sine wave and fit to square wave
Radius of gyration	2	$\mathrm{R}xyz$ and $\mathrm{R}xy$
Radial moment	2	The radial moment and the square of the radial moment
Number of atoms between r and $(r + 1)$ angstroms from the alpha helix axis	10	For $r = 0$ to 9

22.2 TABLE The 39 molecular parameters calculated for each peptide.

that is, it is necessary for a successful peptide to be able to diffuse through an outer cell wall and then form aggregates of peptides according to the barrel-stave model. The ability to diffuse and aggregate is described by two groups of parameters: general molecular parameters and amphipathic descriptors. General molecular parameters include molecular weight, charge, size, and shape parameters. An example of an amphipathic parameter is the hydrophobic dipole moment, a property analogous to the standard electrostatic dipole moment but using hydrophobicity of the amino acids instead of atomic point charges. The x, y, and z components of the dipole are derived by orienting the peptides with z down the center of the alpha helix of the peptide with origin ($z = 0$) at the midpoint; the y-axis is defined as being in the direction of the vector sum over all hydrophobic residues $\Sigma H_i C_i$, where H_i is the hydrophobicity value of Eisenberg, Weiss, and Terwilliger (1982, 1984) and C_i is the coordinates of the $C\alpha$ atom of amino acid i. Electrostatic dipole moments are calculated in a similar manner.

For a peptide to be laterally amphipathic, hydrophobicity should be periodic along the sequence of amino acids, with period ~ 3.6, the circumference of a turn in the alphahelix. Two parameters were devised to approximate to this, the closeness of fit of hydrophobicity to a sine wave, s_{sin}, and the closeness of fit to a square wave, s_{sqr}. These parameters are given by least squares equations:

$$s_{\text{sin}} = \min_\theta \left\{ \sum_1^N [\text{H}_i - \sin(100i - \theta)]^2 \right\} / N$$

$$s_{\text{sqr}} = \min_\theta \left\{ \sum_1^N [0.5 \cdot [\text{sgn}(\text{H}_i) - \text{sgn}(\sin(100i - \theta))]^2] \right\}$$

where N is the number of peptides, i is the position along the amino acid sequence, H_i is normalized such that the largest positive and largest negative value of hydrophobicity for the residues are set at 1 and -1, respectively, and θ is an offset that is varied to find minimum value.

22.4.2 Neural Networks

To build an effective NN model, we employed the following steps: initial data exploration (including linear modeling), parameter selection, choice of model architecture, model training with cross-validation, and blind testing.

Of the 39 initial parameters a smaller subset was selected. The autocorrelation between the input parameters highlighted clusters of similar parameters. Together with the coefficients of the input parameters, calculated by a step-wise linear regression, six key variables were identified that made sense given the mechanism of action:

1. Sum of the negative charge
2. Sum of the hydrophobic and hydrophilic values
3. Closeness of fit to sine wave
4. Closeness of fit to square wave
5. x component of the charge dipole
6. y component of the charge dipole

These represent a selection of general molecular and amphiphilic parameters (i.e., of diffusion and aggregation parameters).

Linear regression on these six variables gives a poor fit, but shows some structure in the plot of residuals. Together with data visualization methods (Kohonen self-organizing map), this indicated nonlinearity in the data. Multi-layer perceptron (MLP) neural networks are described as universal nonlinear approximators and have the ability to model arbitrary nonlinearities (Homick, Stinchcombe, and White 1989), this is, they may be used to find nonlinear correlations between inputs x to outputs y. For a univariate output y, this has a general form:

$$y = \sum_j (w_j(f(\sum_i w_{ij}x_i + b_i)) + b_j)$$

where x_i are the inputs (general molecular and amphiphilic parameters), w_{ij} are the weights between the input layer and the "hidden" layer, w_j are the weights between the "hidden" layer and the single output node, b_i and b_j are the bias weights, or offsets, and f is a nonlinear transfer function, in this case the hyperbolic tangent tanh, though it may be a sigmoid function or a Gaussian.

Classically, training an MLP consists of setting the weights w, such that the mean squared error is minimized over the training patterns. This was done using the delta-bar-delta training algorithm in Neural Works Professional II.[2] A neural network with six hidden nodes was found to give the best results.

As the data set was very small (29 data points), a leave-one-out cross-validation technique was used to ensure the validity of the model. Gathering the predictions across all 29 points, we have derived $R^2 = 0.905$ for the cross-validation, as well as mean $R^2 = 0.968$ across the 29 models.

Recent results suggest that weighted combinations of models may outperform any single model (Bishop 1995). With no a priori reason to preferentially weight any one of the models, the predicted activity was taken to be the mean of the predictions of the 29 models.

---

[2] Neural Works Professional II, NeuralWare, Inc., Penn Center West, Pittsburgh, PA 15276.

22.5 EVOLUTION

The use of genetic algorithms (GAs) in peptide design is particularly apt due to the natural extension of biological evolutionary operators to amino acid representations of peptide sequences (Jones 1994).

There are three important aspects to be determined in the devising of a GA: the representation of the problem; the fitness function or "chance of survival"; and the genetic operators, or method of mating. The problem is naturally represented by the sequence of amino acids, using letters of the alphabet (LLKALL . . .). The fitness function, or method by which survival of any individual is determined, is simply the neural net predictive model: "virtual" peptides that are predicted to be good bactericides survive to produce offspring for later populations.

For this study, standard operators were used: roulette wheel selection, mutation, and two-point crossover. Following a number of short tests on a simplified model, it was noted that the success of the algorithm in finding large numbers of potentially active peptides was robust to the setting of the GA parameters. For a full run, the parameters were set as follows: population size 100, size of elite set 75, probability of crossover 0.6, and probability of mutation 0.033.

For an average run, after 50 generations, 90 out of 100 peptides in the final population were found to be acceptable, that is, with predicted log kill > 7 to both *S. aureus* and *E. coli*, given a total of $50 \times 25 = 1{,}250$ evaluations. This compares very favorably with random generation of peptides, 4 hits out of 52,000 evaluations, and Monte Carlo optimization, 200 evaluations, averaged over 95 runs, to result in a single peptide. Note this efficiency is a minimum comparison: Once the EA converged to a region of high activity, approximately 50% of new peptides generated by the algorithm were found to be active. Comparative results are given in Table 22.3.

Using the GA, over 400 potentially active "virtual" peptides were generated. Five of the 400 were selected to be synthesized by determining the principal components of the six chosen molecular descriptors of the 400 and selecting five peptides that maximized the diversity. The diversity in molecular properties is also reflected in the diversity of the amino acid sequences. The bactericidal properties of these five were measured and are given in Table 22.4.

The measured log kills given in Table 22.4 are given as lower bounds due to a threshold in the measurement sensitivity that is dependent on the initial cell numbers of *S. aureus* bacteria in the test innoculum.

Method	Efficiency
GA	> 90/1,250 = 7.2%
Monte Carlo	1/200 = 0.5%
Random	4/52,000 = 0.008%

22.3 TABLE

Efficiency comparison of the evolutionary algorithm against other standard optimization techniques. The EA is an order of magnitude more efficient.

Peptide Sequence	Predicted Log Kill	Measured Log Kill
AASKAAKTLAKLLSSLLKLL	7.22	> 5.06
LLKKLLRAASKALSLL	7.13	> 5.90
AAKKLSKLLKTLLKLL	7.35	> 5.76
KALKKLLKLASSLLTAL	7.04	5.90
AASKALRTASRSLLTLL	7.03	> 5.85

22.4 TABLE

Experimental measures of log kill of *S. aureus* for peptide sequences that were generated by the genetic algorithm.

22.6 PATENT APPLICATION

To be patentable an invention must be novel, useful, and nonobvious. In addition, it should be described in a manner understandable by a person, or group of people, "skilled in the art." Typically a patent application ends with a number of separate but related "claims." The claim sets out what is covered by the patent, that is, the particular intellectual property that is protected. There are a number of typical conventions in framing patents concerning peptides. A common method is to frame a claim as consisting of a list of individual peptides as given by their amino acid sequences, (Ala-Leu-Thr- . . .) or by abbreviation (LLLKLKKALL . . .). A wider form of patent claim is based on the structural form of the peptide and defines templates in which amino acids may be substituted. A claim may read: peptides of the form "R1-R2-R2-R1-R3- . . . etc." where the R-groups may be defined as hydrophilic amino acids, hydrophobic amino acids, basic hydrophobic, neutral hydrophilic, and so on (Bhakoo, Patel, and Stott 1996; Maloy 1992; Zasloff 1992; Houghten and Blondelle 1994).

A typical example is found in Claim 1 of Zasloff (1992):

> [We claim . . .] a biologically active amphiphilic peptide, said peptide including the following basic structure *x*, wherein *x* is:

> R1-R1-R1-R3-R5-R1-R1-R1-R1-R1-R2-R2-R1-R1-R3-R1-R4-R1-R3-R4-R1-R1
>
> Wherein R1 is a hydrophobic amino acid, R2 is a basic hydrophilic amino acid, and R3 is a neutral hydrophilic amino acid, R4 is a hydrophobic or basic hydrophilic amino acid, and R5 is a hydrophobic, basic hydrophilic, or neutral hydrophilic amino acid.

The scientific justification for this claim is given by experiments conducted on seven peptides:

> GVLSNVIGYLKKLGTGALNAVL
> GVLSKVIGYLKKLGTGALNAVL
> GVLSQVIGYLKKLGTGALNAVL
> GVLSFVIGYLKKLGTGHLNHVL
> GVLSNVIGYLKKLGTGKLNKVL
> GVLSFVIGYLKKLGTGKLNKVL
> GVLSKVIGYLKKLGTGKLNKVL

where these peptides have been shown to be active bactericides, and an implicit inference is made that this bactericidal property will extend to other peptides with a similar hydrophobic—hydrophilic structure.

Let us compare this with Bhakoo, Patel, and Stott (1996). The patent text states:

> We have determined that effective peptides are discriminated from ineffective peptides by means of an equation relating certain properties of the peptides to their biological activity against specific micro-organisms . . . and have demonstrated that a strong correlation is shown between those peptides which satisfy this rule and those which have effective antimicrobial properties. Conversely, it appears that the majority of the peptides which exhibit properties outside the scope of this rule do not show effective antimicrobial properties.

As publishing 29 neural net models would be somewhat cumbersome, for simplicity of publication in the patent, a single neural net model was generated that approximated the same function as the mean of the 29 neural nets. An artificial data set was generated by creating a lattice of $7^6 = 117{,}649$ data points with "observed values" given by the mean of the 29 neural net models. These artificial data were used to train a single neural net model with $R^2 = 0.982$ over the artificial data set. This final model was published in the patent application. Claim 3 of Bhakoo, Patel, and Stott (1996) claims the peptides with specific bactericidal ability to *S. aureus:* "Antimicrobial peptides having a length of 10–30 amino acid residues wherein the predicted log kill to *S. aureus* ATCC 6538,

($L_{S.\ aureus}$) is greater than 5, $L_{S.\ aureus}$ being given by the equation . . ." and there follows several pages of simple algebraic expressions of the neural network, written in a form that may easily be turned into a computer program.

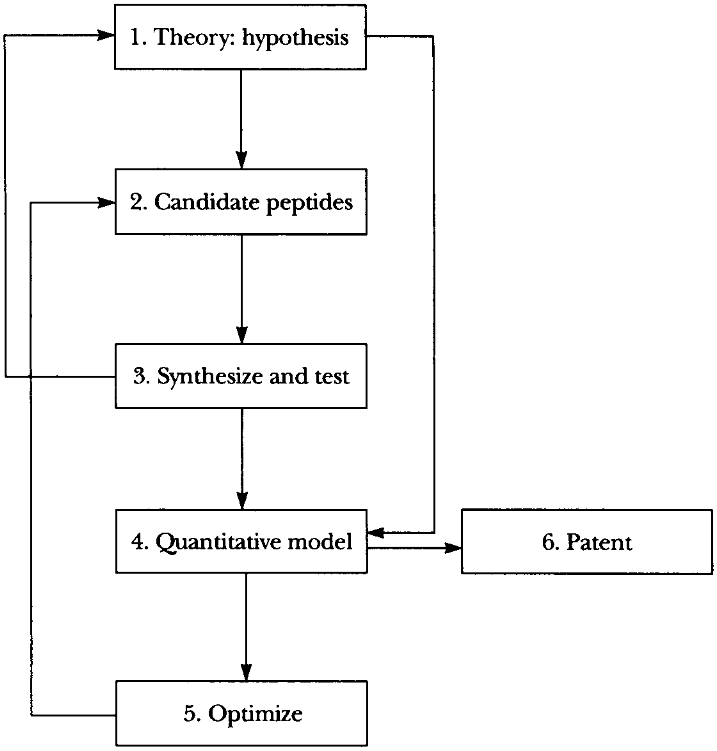
While the neural network and the necessary derivations of the input parameters to the neural network are disclosed in their entirety, the object of the patent is to protect the peptides that the NN predicts to have an activity higher than a specified value. In effect the patent states that we have determined a relationship between peptide 3D structure and bactericidal activity. This rule is given by the neural network model, and is validated by both positive and negative examples. The generalization from specific examples to a wider range of peptides is explicit and is demonstrated to be based on a combination of scientific understanding and experimental evidence. Furthermore, it is substantiated by the success in predicting new peptides, the sequences given in the results, that are indeed active against *S. aureus.* Many thousands of active peptides may easily be found using the model inversion/optimization techniques described here and elsewhere. It is important to note that the neural network is *not* patented or claimed in any way.

In effect we have added a sixth step to the process of computer-aided molecular design—the close interplay of the modeling step with the patenting process (Figure 22.7).

22.6.1 Comparing Patent Spaces

Whether a patent is written in any of the available forms (Bhakoo, Patel, and Stott 1996; Maloy 1992; Zasloff 1992; Houghten and Blondelle 1994), a peptide patent claim may be viewed as defining a region in peptide space; this may be called the *patent space.* It is instructive to analyze the shape and size of a patent space for a particular claim or patent as well as the spread of illustrative examples through this space, and compare across patents. A small example may help to illustrate this point. For a "toy" peptide system consisting of trimers with an amino acid alphabet of {A, B, C} the peptide space consists of a fully connected three-dimensional hypercube. If we take as a hypothetical case a peptide patent that claims R1-R1-R2 where R1 = {A, B} and R2 = {B, C}, the full set of sequences in this patent space is AAB, ABB, BAB, BBB, AAC, ABC, BAC, BBC, and they occupy a connected subcube of the complete space (Figure 22.8).

For Maloy (1992), Zasloff (1992), and Houghten and Blondelle (1994), the shape of the patent space will be a fully connected hypercube nestling within the complete peptide space. The "landscape" defined by a neural network model is more complex, and approaches to studying these are given, for example, in Stadler (1997), Kauffman and Macready (1995), and Levitan (1997). We wish to

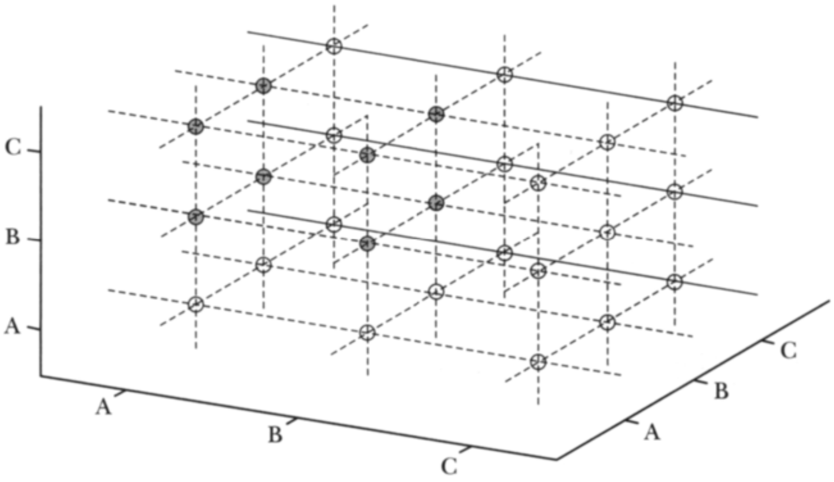


22.7 FIGURE

By using the neural net predictive model to define the scope of a patent claim, we have added a sixth step to the computer-aided molecular design process, the interplay between modeling and patenting.

compare estimates of the relative extent of the experimental space (the diversity of the synthesized peptides) with the extent of the patent spaces claimed. A crude measure of the extent of a peptide (sub-) space is the average Hamming distance between the peptides of the region, where the Hamming distance between sequences S1 and S2 is the smallest number of amino acid substitutions or additions necessary to transform S1 into S2. The average Hamming distance between the points of such a regular subspace may easily be calculated. For a given sequence N amino acids long, with A possible amino acids, the peptides that are s steps away involve $(A - 1)^s$ changes, for which there are ${}_N C_s$ permutations. Summing from 0 to N, the average Hamming distance, D, of the space is given by

$$D = \frac{1}{T} \sum_{s=0}^{N} s \cdot {}^{N}_{s}C (A-1)^{s}$$



22.8

FIGURE

Schematic of a regular subspace of a peptide space of trimers based on an amino acid alphabet of {A, B, C}. The subspace is defined by R1-R1-R2 where R1 = {A, B}, and R2 = {B, C}.

where $T = A^N$ is the total number of peptides in the space. For Maloy (1992), Zasloff (1992), and Houghten and Blondelle (1994), A was estimated at 10, half the typical alphabet of 20 amino acids. For Bhakoo, Patel, and Stott (1996), as there are no structural templates, it was overestimated at 20. H, the average Hamming distance between experimental examples, was calculated directly from the examples for (Bhakoo, Patel, and Stott (1996); Zasloff (1992); Houghten and Blondelle (1994). For Maloy (1992), H was estimated by the substitution method of generating the example peptides described in the patent text.

Table 22.5 shows the comparison of the average Hamming distance between the synthesized peptides given as examples and the average Hamming distance of the patent space they fall within, where n = the number of example peptides within the patent space of the particular claim. The final column gives the comparison between the diversity of the synthesized peptides, and the diversity of the patent space claimed. This value gives a rough measure of the extent to which the experimental results extend across the patent space claimed and may be used as justification for the claim. It can be seen that our patent (Bhakoo, Patel, and Stott 1996) is at least comparable, if not more substantial, in its spread.

Structurally based patents assume that peptide activity will be distributed in a highly regular manner through peptide space. In contrast, a neural net based definition of patent space makes no prior assumptions about the shape of the

Patent	Claim	*n*	*H*	*N*	*A*	*D*	100 × H/D
Maloy (1992) 5294605	7	41	2.03	27	10	16.2	12.5
Zasloff (1992) 92_20358	1	7	2.70	24	10	21.6	12.5
Houghten & Blondelle (1994) 92_17197	1	15	9.45	26	10	24.3	38.9
Bhakoo et al. (1996) 96_28468	3	14	13.20	30	20	28.5	46.3

22.5 TABLE The results of a comparison of the relative spread of synthesized peptides within the patent spaces claimed. *n* = the number of example peptides within the patent space of the particular claim; *H* = the average Hamming distance between experimental examples; *A* = an estimate of the number of possible amino acids in each position in the peptide sequence; *D* = the average Hamming distance between peptides in the peptide patent space. The last column gives an indication of the relative spread within the claim.

claimed region. It is quite possible that there are disconnected regions or even isolated peptides. A more detailed study of the characteristics of these spaces is left to a further study.

22.7 CONCLUSIONS

Computer-aided molecular design has taken a large leap forward in recent years by the use of algorithms, such as neural networks for the prediction of properties of interest, and genetic algorithms that use the neural network models in the discovery and design of new peptides. We have successfully used these techniques to design novel bactericidal peptides. Neural net equations have been determined on the basis of a sound theoretical hypothesis of bactericidal activity, 29 experimental measures, carefully chosen molecular parameters, and the development of a predictive model based on these aspects. Futhermore, this relationship has been validated by its use in computer-aided molecular design; that is, new peptides that are unlikely to have been thought of by conventional methods have been designed by the GAs that were predicted to be active, and synthesis and experimentation on these new peptides have been in accord with the predictions.

The patent aspects of the work have identified several legal issues. It appears reasonable to assume that where new and useful compounds are produced by a method involving intensive computation, those who set up the problem, rather than the computer itself, are the "inventors" of the compounds, despite the fact that the precise nature of the compounds could not have been predicted in advance. We describe how we have used the neural net equations to define the scope of a patent claim. A justification for this approach lies within the context of the well-documented method of computer-aided molecular design. It remains to be seen whether such a mathematical approach will rest easy with the patent offices around the world who must approve the patent for grant.

In the same way that any patent discloses the scientific understanding of a particular scientific endeavour, so we have disclosed the model in full, as well as the precise method for development of the model. The mental act of formulating the claims of a patent, which identifies its scope, has always involved the definition of a broad space within which the examples and many other potential compounds lie. We have begun to show how the diversity of this space can be explored with mathematical tools. In a simple comparison with other patents on bactericidal peptides, we show that in our case the diversity of the synthesized peptides is high. The residues vary in every position, and this chapter presents some evidence to show that they have a relatively wide range within the space of all peptides. While these results are very interesting, considerable further research may be carried out on molecular patent space analysis to address more fully issues surrounding "coverage" of example molecules and claims.

REFERENCES

Bhakoo, M., T. H. Birkbeck, and J. H. Freer (1985). *Canadian Journal of Biochemical and Cell Biology* 63(1).

Bhakoo, M., S. Patel, and I. Stott (1996). Amphiphilic Peptides and Analogs Thereof. WO 96/28468.

Bishop, C. (1995). *Neural Networks for Pattern Recognition.* Clarendon Press.

Christensen, B. (1998). *Proceedings of the National Academy of Science (USA)* 85:5072–5076.

Clark, D. E. and D. R. Westhead (1996). *Journal of Computer-Aided Molecular Design* 10: 337–358.

Degrado, W. F. (1988). *Advances in Protein Chemistry* 39:51.

de Weijer, A. P., C. B. Lucasius, L. Buydens, L. and G. Kateman (1993). In *International Journal of Chemometrics and Intelligent Laboratory Systems* 20:45–55.

Douguet, D., E. Thoreau, and G. Grassy (2000). A Genetic Algorithm for the Automated Generation of Small Organic Molecules: Drug Design Using an Evolutionary Algorithm. *Journal of Computer-Aided Moleculor Design* 14:449–466.

Eisenberg, D., R. M. Weiss, and T. C. Terwilliger (1982). *Nature* 299:371.

Eisenberg, D., R. M. Weiss, and T. C. Terwilliger (1984). *Proceedings of the National Academy of Science (USA)* 81:140.

Fasman, G. (ed.) (1989). *Prediction of Protein Structure and the Principles of Protein Conformation.* Plenum Press.

Freer, J. H., T. H. Birkbeck, and M. Bhakoo (1984). In J. E. Alouf *Bacterial Protein Toxins,* Academic Press, pp. 181–189.

Hancock, I. C., and I. R. Poxton (eds.) (1988). *Bactericidal Cell Surface Techniques.* John Wiley & Sons.

Hellinga, H. W., and F. M. Richards (1994). *Proceedings of the National Academy of Science (USA)* 91:5803–5807.

Hornick, K., M. Stinchcombe, and H. White (1989). *Neural Networks* 2:359–366.

Houghten, R. A., and S. Blondelle (1994). Amphiphilic Peptide Compositions and Analogs Thereof. U.S. Patent 5,294,605. Filed July 8, 1991. Issued March 15, 1994.

Jin, A. Y., F. Y. Leung, and D. F. Weaver (1997). *Journal of Computational Chemistry* 18(16):1971–1984.

Jones, D. T. (1994). De Novo Protein Design Using Pairwise Potentials and a Genetic Algorithm. *Protein Science* 3:567–574.

Kauffman, S. A., and W. G. Macready (1995). *Journal of Theoretical Biology* 173:427.

Levitan, B. (1997). *Annual Reports in Combinatorial Chemistry and Molecular Diversity* 1:95.

Maloy, W. L. (1992). Novel Peptide Compositions and Uses Thereof. WO 92/17197.

Maranas, C. D. (1997). *American Institute of Chemical Engineers (AIChE) Journal* 43(5):1250–1264.

Mavrovounitis, M. L. (1996). In J. F. Davis, G. Stephanopopoulos, and V. Venkatasubramanian (eds.), *Proc. Int. Conf. on Intelligent Systems in Process Engineering,* vol. 92, AIChE Symposium Series no. 312, p. 133.

Patel, S., I. P. Stott, M. Bhakoo, and P. Elliott (1998). Patenting Computer-designed Peptides. *Journal of Computer-Aided Molecular Design* 12:543–556.

Parrill, A. (1997). *Journal of Expert Opinion Therapeutic Patents* 7(9):937–945.

Schneider, G., M.-L. Lee, M. Stahl, and P. Scheider (2000). De Novo Design of Molecular Architectures by Evolutionary Assembly of Drug-Derived Building Blocks. *Journal of Computer-Aided Molecular Design* 14:487–494.

Schneider, G., J. Schuchhardt, and P. Wrede (1995a). *Biological Cybernetics* 73(3).

Schneider, G., J. Schuchhardt, and P. Wrede (1995b). Peptide Design in Machina: Development of Artificial Mitochondrial Protein Precursor Cleavage Sites by Simulated Molecular Evolution. *Biophysical Journal* 68:434–447.

Stadler, P. F. (1997). Santa Fe Institute paper 97-11-082, Santa Fe Institute.

Veld, G. I., A. J. M. Drissen, and W. N. Konings (1993). *Federation of European Microbiological Societies (FEMS) Microbiology Reviews* 12:293.

Venkatasubramanian, V., K. Chan, and J. M. Caruthers (1994). *Computers and Chemical Engineering* 18(9):833–844.

Venkatasubramanian, V., K. Chan, and J. M. Caruthers (1995). Evolutionary Design of Molecules with Desired Properties Using the Genetic Algorithm. *Journal of Chemical Information and Computer Sciences* 35:188–195.

Venkatasubramaniam, V., A. Sundaram, K. Chan, and J. M. Caruthers (1994). *Abstract Papers of the American Chemical Society* 207, 60-COMP part 1, p. 396.

Wade, D., D. Andreu, S. A. Mitchell, A. M. V. Silveira, A. Bowman, H. G. Bowman, and R. B. Merrifield (1992). *International Journal of Peptide and Protein Research* 40:429–436.

Zasloff, M. A. (1992). Novel Biologically Active Peptide Compositions and Uses Thereof. WO 92/20358.

This Page Intentionally Left Blank

Index

A

"Aaron," 395
AASPEM (Air-to-Air System Performance Evaluation Model), 474, 481
Abd-El-Barr, M. H., 450
Absolut Vodka images, 235
Adami, Chris, 94, 105
adaptation, natural model for, 255
Adaptation in Natural and Artificial Systems, 280
adaptive artworks, 235–237
Adewuya, A., 494, 495
ADFs (automatically defined functions) in GP, 19–20, 49
ADIs (automatically defined iterations) in GP, 20, 49
ADLs (automatically defined loops) in GP, 20, 49
Adobe PhotoShop, 403
ADRs (automatically defined recursions) in GP, 20
ADS (automatically defined stores) in GP, 20
Advances in Genetic Programming series, 281
aesthetic evaluation
  beauty as measure of quality, 125
  human guidance for, 39–41, 42
  See also evaluation
aesthetic objects, breeding. *See* breeding aesthetic objects
Afman image, 341–347
  genealogical tree of, 343–344
  generation of, 341, 342
  genomes, 343, 345, 346, 347
  illustrated, 343
  parents of, 341, 343
  starting population for, 341, 342
AI. *See* artificial intelligence (AI)
aiding creativity, 39–42
  collaborative evolutionary algorithms, 40–42
  role of human guidance in, 39–41
Air-to-Air System Performance Evaluation Model (AASPEM), 474, 481
Alberti, Leon Battista, 125
Algorists, 362
algorithmic images, genetically evolved. *See* art using GP
algorithms
  for circuit assembly from components, 447–450
  collaborative evolutionary algorithms, 40–42
  in electronic art movements, 379–380
  EP, 24, 25, 26
  ES general algorithmic scheme, 22
  genetic (GAs), 8, 10–16
  local search, 6

algorithms *(continued)*
  Sound Gallery, 237–242
  in Western music, 206
  See also evolutionary algorithms (EAs); evolutionary programming (EP); genetic algorithms (GAs); genetic programming (GP)
Alive Art, 379–384
  Alive Art effect, 382, 383–384
  Alivers, 381–382
  Artificial Painter, 375–378
  defined, 380–381
  historical context of, 379–380
  LEGO MINDSTORMS robots, 383–384
Alivers, 381–382
Alpern, A., 233
alphabet of L-systems, 300, 302, 303
alphahelical peptides, 528
  See also evolved bactericidal peptides
Altshuler, E. E., 493
amphiphilic peptides, 528, 529
  See also evolved bactericidal peptides
amplifier circuit synthesis using GP, 288–289
analog electrical circuit synthesis using GP. *See* circuit synthesis applications of GP
analysis tools vs. design tools, 316–317
AND circuit symbols, 449
AND gates, 446, 447, 449, 450, 451, 454, 455
Andre, David
  chapter by, 275–298
  See also biologically inspired evolutionary computation
animal behavior, imitation in, 156
animats, 375–378
antenna design using GA, 487–510
  antenna basics, 489–493
  antenna pattern, 490
  bandwidth, 491
  beamwidth, 491, 493, 497
  classes of antennas, 489
  conclusions, 509–510
  crooked-wire genetic antennas, 501–506
  directive antennas, 491
  directivity, 490, 507
  gain, 490, 493, 494, 502, 507
  gain pattern, 490–491, 498
  ground plane, 490
  inductive process vs., 487–489
  multipath problems, 501
  Numerical Electromagnetics Code, Version 2 (NEC2), 490, 492–493
  optimization problems, 487–488
  parasitics, 490
  patent for antennas, 504, 506
  polarization, 492, 494, 502
  for satellite-to-ground communications, 500–509
  for situations with little information, 489
  treelike genetic antennas, 506–509
  Voltage Standing Wave Ratio (VSWR), 491–492
  wire antenna overview, 489–493
  wire size, 492, 507, 509
  Yagi-Uda antenna, 493–500
  See also crooked-wire genetic antennas; treelike genetic antennas; Yagi-Uda antenna
antenna pattern, 490
architects, creative evolutionary systems research by, 37
architecture design. *See* Intelligent Genetic Design Tool; Lindenmayer systems for architecture design in GP
architecture for self-reproduction (von Neumann)
  embeddedness in, 94
  overview, 82–84
  richness of interactions in, 96
  Tierra-like systems analyzed in terms of, 87–89
architecture of GenJam, 166–168
Arecibo spherical reflector (Puerto Rico), 493–498, 506
Argenìa, 109–127
  Argenìc art, 123, 124
  Argenìc design, 123
  artificial DNA in, 111–112, 114
  Basilica project, 116, 118–119
  Borghetto Flaminio project, 116, 117
  coining of term, 109
  complexity management in, 110, 111–114, 116
  conclusions, 125
  control structure of recognizability in, 110
  creativity, approach in, 110
  design decision cycle in, 111–112
  as designer of species, 111
  formal matrices in, 112
  generative approach of, 109, 110
  industrial object design in, 121–123, 124
  logic structures in, 111, 118
  medieval town (a la Giotto) project, 114–116
  metaprojects (idea-projects) with, 109
  multimedia urban stand (Milan) project, 119, 121

  operable fields for margins in, 112
  patinas in, 112
  Picasso repainting with, 123, 124
  Prado Museum (Madrid) project, 119, 120
  recognizability in, 109–110, 125
  species use in, 110, 111, 113, 114
Arnolfini gallery, 130
art
  electronic, 372–378
  evolved images as, 360–364
  feedback to technology from, 370
  found vs. created, 360–361
  genetic cross dissolve, 358–359
  genetic fractals, 351–358
  GP for, 339–351
  historical evolution of, 369
  movements based on electronics, 379–380
  paradigm shift, computational arts and, 388
  technological influences, 369–370
  theory and, 388–389
  See also art using GP; electronic art; Escher Evolver project; evolutionary generation of faces
Art and Aesthetics of Artificial Life exhibit, 364
Art as the Evolution of Visual Knowledge, 391
art gallery, as Sound Gallery setting, 230
art using GP, 339–351
  Afman image, 341–347
  breeding ratios, 348
  cloning, 348
  constraints of color and form, 361–363
  Crossing, The, 349
  Darwin meets Dali, 349
  evolved images as art, 360–364
  found vs. created art, 360–361
  functions and node internals, 347–348, 349
  genetic cross dissolve, 358–359
  genetic fractals, 351–358
  genetic library, 344–347
  genetic variation, 340–344
  genomes, 340, 343, 345, 346, 347, 348
  horizon lines and fantasy landscapes, 351, 352
  mutations, 340–341
  organic images, 364
  postprocessing images, 362
  precedents and first steps, 339–340
  printmaking, 350–351
  typical run, 348–351
  visual cortex responses, 363–364
  See also Afman image; genetic fractals
artificial DNA in Argenia, 111–112, 114
artificial intelligence (AI)
  creativity research in, 2–4
  grammars, 3
  knowledge-based systems (KBSs), 3
  recommended readings, 63–64
  requirements for, 276–277
  search, 3–4
artificial life models
  animats in, 375–377
  Artificial Painter and, 375
  definitions of "life," 86, 91
  energy modeling in, 99–100
  interactions needed in, 91–92
  lack of knowledge for, 86
Artificial Painter, 375–378
  animats, 375–378
  overview, 375
  technology's affect on art and, 372
  as tool for immaterial artists, 375
  user guidance in, 377
Artist view of creativity, 62
artists
  Algorists, 362
  Alivers, 381–382
  Brown's approach, 387–389
  creative evolutionary systems research by, 37–38
  cybernetics, 380
  electronic, 373–375, 379–380
  evolographers, 361
  fitness function role of, 51
  hardware, 379
  immaterial, 370–371, 374–375
  material, 371–372, 373–374
  role in evolutionary process, 133–136, 144
  software, 379
aspect angle for aircraft, 469–470
associations, arbitrary vs. meaningful, 153–154
asterisk (*) as don't care symbol, 29
Astrom, K. J., 291
atheist view of creativity, 62
Austrian Arts Electronica festival, 130
automated design, 511
  See also physical robotics
automatically defined functions (ADFs) in GP, 19–20, 49
automatically defined iterations (ADIs) in GP, 20, 49

automatically defined loops (ADLs) in GP, 20, 49
automatically defined recursions (ADRs) in GP, 20
automatically defined stores (ADS) in GP, 20
autonomous artworks, 235–237
autonomous robots
  design issues, 511–512
  musicology benefits from, 189
  See also physical robotics
Avida platform, 94, 95
axiom of L-systems, 300

B

Bach, C. P. E., 206
Bach, J. S., 372
Bachiana No. 4, 210
Bäck, Thomas, 8, 31
bactericidal peptides. *See* evolved bactericidal peptides
bandwidth in antenna design, 491
Banzhaf, W., 281
Barratt, C. H., 291
Barricelli, Nils, 80, 91–92, 103
Barto, A. G., 486
Basilica project in Argenìa, 116, 118–119
beach, as Sound Gallery setting, 229
beamwidth in antenna design
  defined, 491
  Yagi-Uda antenna, 493, 497
Beardsley, Aubrey, 259
beauty, as measure of quality, 125
beer can theory of creativity, 147–161
  associations, arbitrary vs. meaningful, 153–154
  breadth-first vs. depth-first exploration, 152
  conclusions, 158–159
  creativity as origin of culture, 154–156
  culture as an evolutionary process, 148–149
  evolution of ideas, cultural requirements for, 150
  Meme and Variations (MAV) computer model, 151–152
  metaphor explained, 147
  onset of creativity, causes of, 156–158
Belouzow-Zhabotinsky-style chemical reactions, 196
Benjamin, Walter, 369
Bennett, Forrest H., III
  chapter by, 275–298
  See also biologically inspired evolutionary computation
Bentley, Peter J., 33, 34, 35, 38, 39, 41, 45, 60
  chapter by, 1–75
  genetic algorithm designer (GADES) of, 41–42, 56–57, 58
Beowulf supercomputer, 39
Bhakoo, Manmohan, 538, 541
  chapter by, 525–545
  See also evolved bactericidal peptides
Biederman, Charles, 391
Biles, John A., 163, 235
  chapter by, 165–187
  See also GenJam
binary circuit symbols, 448–449
biological control in Vox Populi, 216
biologically inspired evolutionary computation, 275–298
  circuit synthesis applications of GP, 283–290
  for circuit topology, sizing, placement, and routing, 288–290
  conclusions, 296
  controller synthesis applications of GP, 290–294
  criteria for human-competitive results, 277
  elliptic lowpass filter application of GP, 287–288
  GP overview, 280–282
  human-competitive results from GP, 277–280
  illogical nature of creativity and evolution, 294–296
  ladder filter application of GP, 285–286
  M-derived half section circuit application of GP, 286–287
  problem statements for, 275–276
  two-lag plant controller application of GP, 292–294
biology
  creativity metaphors from, 148
  variation and convergence in, 148–149
  See also natural evolution
Biomorphs program, 130, 139, 316
Black, Harold S., 295
Blind Watchmaker, The, 130, 316, 339
bloat in GP, 19
Blondelle, S., 539, 541
Boden, M. A., 61
Bolton, R. J., 450

Booij, I.
  chapter by, 425–439
  See also Escher Evolver project
Book of Changes, 391
Boolean algebra, 445–446, 447, 448, 449
Borenstein, J., 231
Borghetto Flaminio project in Argenìa, 116, 117
Born, J., 31
Box, George E. P., 9
Boyd, S. P., 291
Braun, F., 487
breadth-first exploration, 152
Breed project, 139–141
  artificial genome in, 140
  evolutionary computation in, 140–141
  morphogenesis in, 140
breeding
  algorithmic images, photography compared to, 361
  creation and, 133–137
  as creative modification, 39
  ratios in GP art, 348
breeding aesthetic objects, 129–145
  approach of Driessens and Verstappen, 139–144
  Biomorphs program, 130, 139
  Breed project, 139–141
  computer science and technological culture, 129–130
  conclusions, 144
  creation and breeding, 133–137
  evaluation of results, 136–137
  evolved aesthetic objects, 136–137
  historical lineage for, 130–131
  limits, 17–19
  Rooke case study, 131–133
  Tuboid sculptures project, 142–144
Briscoe, Chris, 395
Brown, C. T., 80
Brown, Paul, 338, 387–407
  approach as an artist, 387–389
  cellular automata, 396–400
  conclusions of, 405–406
  current and future directions of, 405
  early computer work of, 396–402
  evolving time-based system, 405, 406
  historical work of (1960s and 1970s), 392–396
  major influences on, 389–392
  recent work of, 402–404
  tile drawings, 393–395
building block hypothesis, 30
building blocks, 30
Burks, A., 84

C

CA. *See* cellular automata
CABs (configurable analog blocks), 227–228
Cage, John, 206, 232
Cairns-Smith, Graham, 90, 93
Caldwell, C., 412
Callender, Albert, 291
Callender and Stevenson 1939 PID controller patent, 291
CAMD (computer-aided molecular design), 525–526, 527, 528, 542
  See also evolved bactericidal peptides
Campbell 1917 ladder filter patent, 285–286
Campbell, George, 285
canonical (simple) GAs
  ES vs., 20–21
  in example creative application, 52
  overview, 12–14
Carnegie Mellon MIDI Toolkit, 166
Cauchy mutation in EP, 32
Cauer 1934–1936 elliptic filter patents, 287–288
Cauer, Wilhelm, 287
CCNR (Centre for Computational Neuroscience and Robotics), 242
CD-fit, 409, 414
cell states in ChaOs system, 197–198
cellular automata
  basics, 195–196
  in Brown's work, 396–400
  in ChaOs system, 196–198
  Conway's "Game of Life," 396
  epigenetic development and, 267, 268
  evolving sound with, 194–201
  for granular synthesis, 195
  motes compared to, 267
  musicology benefits from, 189
  in Sound Gallery, 242, 243
cellular encoding, explicit embryogeny in, 50
Centre for Computational Neuroscience and Robotics (CCNR), 242
Centre Georges Pompidou (Paris), 131
Chakrapani, J., 212
ChaOs system, 196–202
  behavior specification, 198
  cell states, 197–198
  cellular automaton (CA), 196–199

ChaOs system *(continued)*
  commentary on the results, 201–202
  global transition function, 198
  mapping technique, 199–201
  metaphor behind, 197
  overview, 196–197
  sound granules, 199–201
  synthesis engine, 199–201
  tone color variations, 201–202
chase chorus with GenJam, 168, 179–184, 185–186
CHC method in IGDT, 324–328
Chellapilla, K., 25, 27, 33
Chomsky, N., 380
chord progression file for GenJam, 166–167
chord-scale mappings for GenJam, 169, 170
choruses file for GenJam, 167
Chromos, 402, 403
chromosomes
  as adaptive structures, 255
  for circuit design discovery using GA, 448
  in IGDT, 324
  MIDI, in Vox Populi, 209
  schema for, 29
  strategy parameters in ES, 21
  in treelike genetic antenna GA, 506
  in Yagi-Uda antenna GA, 494
circuit design discovery using GA, 443–466
  2-into-1 subcircuit principle, 462–464
  6–6–6 subcircuit principle, 462
  6–6–7 subcircuit principle, 462, 464
  6–6–10 subcircuit principle, 462, 464
  algorithms for assembly from components, 447–450
  AND gates, 446, 447, 449, 450, 451, 454, 455
  assemble-and-test method, 446–447
  binary circuit symbols, 448–449
  blind evolution vs. human design, 443
  chromosomes, 448
  conclusions, 464–465
  crossover operator, 448
  fingerprinting, 445, 462–464
  MAX gates, 450
  MIN gates, 450
  MODPRODUCT gates, 450
  MODSUM gates, 450
  modular construction and, 444
  multiple-valued circuit symbols, 449–450
  multiple-valued one-digit adder with carry, 459–461
  MUX gates, 446, 449, 451–452
  NOT gates, 446, 447, 450
  objective, 444
  one-bit adder, 450–452
  OR gates, 446, 447, 450, 451
  principle extraction, 443, 444–445, 462–464
  results, 450–461
  ripple-carry principle, 444, 452
  space of all representations, 445–447
  T-GATEs, 450, 461
  The Fundamental Question (TFQ), 444, 457
  three-bit multiplier, 457–459
  TPRODUCT gates, 450
  TSUM gates, 450
  two-bit adder, 452–454
  two-bit multiplier, 454–457
  XOR gates, 449, 450, 452, 457
circuit synthesis applications of GP, 283–290
  circuit-constructing program tree functions, 284
  elliptic lowpass filter, 287–288
  ladder filter, 285–286
  M-derived half section, 286–287
  overview, 283–285
  preparatory steps, 285
  for topology, sizing, placement, and routing, 288–290
circular polarization in antenna design, 492, 502
Cliff, D., 80
cloning, in GP art, 348
Coates, Paul, 37, 50
cobra maneuver, 470
code scripts in generative evolutionary paradigm, 255, 257
coevolution
  coevolutionary processes, 80
  in generative evolutionary paradigm, 269
  of LCSs, 472
  musical evolution and, 192
  physical robotics and, 512–513, 520
Cohen, Harold, 395
coherence, self-organization and, 192
collaborative evolutionary systems
  advantages and disadvantages of, 41
  defined, 10
  evolutionary operation as, 10
  role of human guidance in, 39–41
collaborative GAs, 15
collapsed cell state in ChaOs system, 197, 198
"Colloquy of Mobiles, The," 235

colors
  constraints for evolved images, 361–363
  in Escher Evolver project, 429, 431
competition for limited resources
  embeddedness in arena of, 93–98
  fitness functions and, 80–81
  fundamental resources, 98–99
  predator-prey relationships and, 98
complexity
  Argenìa management of, 110, 111–114, 116
  coevolution and, 192
  in musical evolution, 192
  organic forms and, 364
  in Rooke's work, 133
  theory, wide applicability of, 154
component-based representations, 44–46
  design problems for creative evolution, 48–49
  example creative application, 52–54
  exploration, 45–46, 48, 49
  optimization, 44–45
computational circuits, GP synthesis of, 288–289
computational metamedium, 388
computer art, 379
Computer Arts Society, 395
Computer Artworks, 130
computer science, technological culture and, 129–130
computer-aided molecular design (CAMD), 525–526, 527, 528, 542
  See also evolved bactericidal peptides
"Computing Machinery and Intelligence," 282
concept seeding approach in generative evolutionary paradigm, 259–260
conceptual arts, 371
conditional statements in GP, 19
configurable analog blocks (CABs), 227–228
"Conquest of Form, The," 130
Conrad, M., 80, 101, 103
consonance, 213
consonance criterion in Vox Populi, 212–214
constraints
  parameterization and, 43–44
  removal of, 43–44
construction vs. description, 84
constructive machine, 83
continuous EP, 26–27
control automaton, 83
control structure of recognizability in Argenìa, 110
controller synthesis applications of GP
  overview, 290–292
  for two-lag plant, 292–294
convergence
  in culture and biology, 148–149, 158
  as evolution requirement, 149
  GenJam issues, 170–172, 178–179
  premature, 14–16
convergent evolution by natural selection, defined, 255–256
Conway, John Horton, 262, 392, 396
coordinate systems for Universal State Space Modeler, 265–266
copying
  schema theorem and, 30
  in Vox Populi, 209
copying automaton, 83, 84
Corne, David W., chapter by, 1–75
cost function for crooked-wire genetic antenna, 502
covariance, 27–28
covariance and selection theorem
  building block hypothesis and, 30
  future generations and, 30
  overview, 27–29
Craenen, B., 426
Cramer, N. L., 16
creative agency, breeding process and, 133–136
creative evolutionary design
  Bentley's definition of, 60
  breeding process and creative agency, 133–136
  momentum of process, 132, 135
  See also biologically inspired evolutionary computation; generative evolutionary paradigm; Intelligent Genetic Design Tool; Lindenmayer systems for architecture design in GP
creative evolutionary systems
  for aiding our creativity, 39–42
  background, 36–39
  defined, 36
  example application, 52–54
  framework for, 46–52
  for generating creative results, 42–46
  goals of, 36
  human guidance and, 38
  representations as primary for, 48
creative process, Ehrenzweig's stages of, 390

creativity, 54–62
  AI research, 2–4
  biological metaphors for, 148
  as clock of evolution, 110
  computer science use of term, 1
  connotations of, 55
  correlation with intelligence, 159
  definitions of, 57–62
  in design, 253
  Ehrenzweig's stages of the creative process, 390
  evolution as creative, 56–57, 58–62
  found vs. created art, 360–361
  of GenJam, 184
  as going beyond the bounds of representation, 61–62
  illogical nature of, 294–296
  innovation vs., 60–61
  inventiveness and, 276–280
  linkage disequilibrium and, 153
  machine-assisted, 271–272
  in natural evolution, 2
  nonrandomness of, 152
  in "novel solutions that are qualitatively better than previous solutions," 59
  Nureyev's definition of, 148
  open-ended evolution and, 79–82
  as origin of culture, 154–156
  overview, 55
  recommended readings, 63–64
  as removal of constraints, 43
  in search space exploration, 60
  as soul expression, 62
  subjective, 62
  in "surprising and innovative solutions," 58–59
  test pilot creativity simulation, 467–486
  theory and, 388–389
  in transferring information from other domains, 60–61
  as universal in humans, 148, 158
  in working with little or no information, 59–60
  See also Argenia; beer can theory of creativity; breeding aesthetic objects; open-ended evolution; test pilot creativity simulation
crooked-wire genetic antennas, 501–506
  circular polarization, 502
  conclusions, 509
  cost function, 502
  distribution of randomly selected antennas, 504, 505
  gain, 502
  goal, 502
  other configurations, 504, 505
  patent for, 504, 505
  search space, 501, 502
  seven-wire configuration, 502–504
cross dissolve, genetic, 358–359
Crossing, The, 349
crossover operator
  for circuit design discovery using GA, 448
  in GAs, 12, 13
  in generative evolutionary paradigm, 255, 258
  in GenJam, 173–175
  in GP, 17, 18, 31
  N-point crossover in Escher Evolver project, 432–433
  schema theorem and, 30
  in Vox Populi, 209
  in Yagi-Uda antenna GA, 494
culture
  as breadth-first social matrix, 152
  creativity as origin of, 154–156
  emergence of, 157–158
  evolution of ideas and, 150
  as evolutionary process, 148–149
  genotype analog in, 149
  historical origins of, 155
  mental set and evolution in, 153
  phenotype analog in, 149
  self-organization in, 193–194
  as self-sustained system, 149
  technological, computer science and, 129–130
  variation and convergence in, 148–149, 158
  worldview and, 157–158
curves in Escher Evolver project, 429, 430, 431
Cybernetic Serendipity, 392
cybernetics artists, 380

D

Dancer DNA, 131
Dannenberg, Roger, 166
Darwin, Charles, 98, 256
Darwin meets Dali, 349
Dawkins, Richard, 37, 39, 130, 133, 135, 136, 142, 233, 316, 339
decibels of gain in antenna design, 490
Deeper Blue, 4

defining length of schema, 29
depolarized cell state in ChaOs system, 197, 198
depth-first exploration, 152
description vs. construction, 84
design
  designer's vs. historian's perspective, 253
  evolutionary system advantages in, 317–319
  imitation of human process, 319
  as problem-finding activity, 253
  Soddu's definition of, 110
design decision cycle in Argenìa, 111–112
design fixation, removing, 43
design issues
  ability to perform other tasks, 91–93
  architects' approach to, 37
  for architectural design, 320–321
  artists' approach to, 37–38
  for autonomous robots, 511–512
  for circuits, 283
  designers' approach to, 37
  embeddedness of individuals and richness of interactions, 93–98
  human element for, 36–37
  implicit vs. explicit encoding, 87–91
  materiality, 98–100
  for open-ended evolution, 82–100
  in Tierra platform, 84–87
  in von Neumann's architecture for self-reproduction, 82–84
design schemas, 259
design space for treelike genetic antennas, 507
design tools
  analysis tools vs., 316–317
  GAs for, 315–316
  See also Intelligent Genetic Design Tool
Deutsches Museum (Munich), 130
deviation parameters for mutations in ES, 21
Dice Game ("Wurfelspiel"), 206
digital amber genomes, 132–133
digital art, 379
digraph genomes, 348
Dike, B. A.
  chapter by, 467–486
  See also test pilot creativity simulation
diploidy, GAs with, 15
directive antennas, 491
directivity in antenna design, 490, 507
Director, 402
directors in Yagi-Uda antenna, 493
disruption in GP, excessive, 20
distributed agents, 190
distributed GAs, 15
divergent evolution, 256
DNA, phenotypes and, 271
do Valle, Raul, 210
dominance
  GAs with, 15
  GP and, 19
don't care symbols, 29, 472
Driessens, Erwin, 139–144
drift, 151
dual-goal evolution in L-systems, 309
Duchamp, Marcel, 371
duration, 208

E

EAs. *See* evolutionary algorithms (EAs)
Ebcioglu, K., 3
Echo model, 97
editing operator in GP, 19
efficiency of evolution-based algorithms, 9
Ehrenzwieg, Anton, 389–390, 393
Eiben, A. E., 338, 426
  chapter by, 425–439
  See also Escher Evolver project
eigenfaces, 409–412
  basis set for, 409–410
  blurring of, 410
  eigenshapes for, 411–412, 413
  normalized and cropped faces for, 416
  PCA production of, 409
  recombination of, 410–411
  See also evolutionary generation of faces
eigenshapes, 411–412, 413
Einstein, Albert, 158–159, 318
Eizenberg, J., 3
electrical circuits. *See* circuit design discovery using GA; circuit synthesis applications of GP
"Electroacoustic Samba X," 202
electromechanical systems, 517
electronic art, 372–378
  Alive Art, 379–384
  art movements based on electronics, 379–380
  Artificial Painter, 375–378
  cybernetics artists, 380
  hardware artists, 379
  by immaterial artists, 374–375
  LEGO MINDSTORMS robots, 383–384

electronic art *(continued)*
  by material artists, 373–374
  material vs. immaterial artists, 370–372
  social context of electronics, 368–370
  software artists, 379
  space for, 373, 374
electronic space, 373, 374
electronics design
  artificial evolution for, 224–226
  See also circuit design discovery using GA; circuit synthesis applications of GP
El-Fallah, A.
  chapter by, 467–486
  See also test pilot creativity simulation
Elliott, Peter
  chapter by, 525–545
  See also evolved bactericidal peptides
elliptic lowpass filter application of GP, 287–288
elliptical polarization in antenna design, 492
embeddedness, 93–98
  in architecture for self-reproduction (von Neumann), 94
  in Avida platform, 94, 95
  creative evolutionary possibilities and, 96
  open-ended evolution and, 93–98
  richness of interactions and, 94–96
  in Tierra platform, 94, 95
embryogeny
  in creative evolution framework, 49–50
  in example creative application, 52, 53
  explicit, 50
  external, 50
  genotypes vs., 51
  implicit, 50
  phenotypes vs., 51
embyrology in L-systems, spatial, 302
emergent properties of EAs, 6–7
encapsulation operator in GP, 19, 49
encoding scheme in Sound Gallery, 240
energy, modeling in artificial life models, 99–100
environment
  complexity management in Argenìa, 110, 111–114
  evolutionary processes and, 100–101
  materiality, 98–100
  phenotype interaction with, 88
  selection for "mirror image" of, 149
EP. *See* evolutionary programming (EP)
epigenetic development, 255, 257, 267
epistasis, 151
E-plane of Yagi-Uda antenna, 493, 497, 498
ES. *See* evolution strategies (ES)
Escher Evolver project, 425–439
  colors, 429, 431
  conclusions, 437–438
  curves, 429, 430, 431
  essential aspects of, 427
  evolutionary algorithm, 428–433, 436
  filling shapes, 429, 431, 432
  first networked version, 434–435
  flatfish example, 433
  Gaussian mutation, 433
  genetic operators, 432–433
  ground shapes, 427, 430
  implementation, 434–437
  mathematical system behind Escher's tiling, 427–428
  N-point crossover, 432–433
  origins of, 425–427
  plane transformations, 427–428, 429, 432
  possible combinations of shapes and transformations, 427–428
  representation, 429
  second networked version, 435–437
  selection mechanism, 433
  stand-alone version, 434
  user interface, 435, 436–437
  visitors' reactions, 435, 437
Escher, M. C., 425, 427
Eshelman, L., 324
Euro-GP conference, 281
evaluation
  aesthetic, human guidance for, 39–41, 42
  beauty as measure of quality, 125
  in EA design space, 34
  of evolved aesthetic objects, 136–137
  in generative evolutionary paradigm, 255, 257–258
  of GenJam, 168
  in GP, 280
  mapping vs., 34
  in Reptile system, 264
  of Vox Populi, 211–216
  See also fitness; fitness functions; human guidance
evolographer, 361
evolution
  divergent, 256
  Gabora's definition of, 148
  of ideas, 149, 150
  illogical nature of, 294–296
  Miranda's usage of, 190–191
  requirements for, 148–149
Evolution of Communication Journal, 190

evolution of ideas, cultural requirements for, 150
evolution strategies (ES), 20–24
  (μ +)-ES, 23
  (μ,)-ES, 23–24
  1/5 success rule, 31
  advanced, 23–24
  approximation theory for convergence rate, 31
  for circuit design discovery, 448, 457
  component-based exploration and, 48
  development of, 8, 20
  deviation parameters for mutations, 21
  evolutionary computation theory, 31
  fitness evaluation, 22
  GAs vs., 20–21, 23
  general algorithmic scheme, 22
  mutations in, 21, 22, 23
  population-oriented, 22
  recommended readings, 64–65
  self-adaptation in, 23
  strategy parameters, 21, 22, 23
evolutionary algorithms (EAs)
  collaborative, 40–42
  component-based exploration in, 45
  in creative evolution framework, 47–48
  in Escher Evolver project, 428–433, 436
  evolutionary biology and, 8
  fitness evaluation in, 7–8
  fitness value curve, 32–33
  generalized (GAEA), 33–35
  guidance in, 7–8
  importance of, 9
  main families of, 8, 33
  music and, 232–234
  musicology benefits from, 189
  overview, 5–7
  solutions as emergent property of, 6–7
  visual arts and, 231–232
  See also evolution strategies (ES); evolutionary programming (EP); genetic algorithms (GAs); genetic programming (GP)
"Evolutionary Art Bottle Breeder," 235
evolutionary computation
  biologically inspired, 275–298
  in *Breed* project, 140–141
  as creative, 56–57, 58–62
  design problems for creative evolution, 48–49
  historical development of, 9–10
  natural evolution as mentor for, 2
  overview, 4–9
  theory, 27–33
  See also biologically inspired evolutionary computation
evolutionary computation theory, 27–33
  1/5 success rule, 31
  approximation theory for convergence rate in ES, 31
  building block hypothesis, 30
  covariance and selection theorem, 27–29
  EP theory, 31–32
  ES theory, 31
  fitness value curve, 32–33
  natural evolution vs. computer evolution, 27
  schema theorem, 29–31
Evolutionary Design by Computers, 251, 337
evolutionary generation of faces, 409–423
  apparatus, 415
  conclusions, 42–43
  discussion of results, 421–422
  eigenfaces, 409–412
  eigenshapes for, 411–412, 413
  evolutionary algorithm, 416–417
  evolutionary face generator system, 412–414
  generation of face images, 415–416
  hair omitted in, 422
  MSE as performance measure, 418–421
  normalized and cropped faces for, 416
  participants, 417
  pilot system, 414
  rating faces, issues for, 421–422
  results, 417–421
  target faces, 416, 417
  testing, 415–421
  user interaction, 412, 414
evolutionary modeling, 190–194
  coevolution, 192
  fundamental mechanisms for studying origins of language, 190
  level formation, 194
  self-organization, 192–194
  transformation and selection, 191–192
evolutionary musical cycle of Vox Populi, 208–211
  genetic cycle, 209–210
  rhythm of the evolution, 210–211
  voices population, 209
evolutionary operation, 9–10
evolutionary paradigm. *See* generative evolutionary paradigm
evolutionary programming (EP), 24–27
  advanced, 25–27

evolutionary programming *(continued)*
  algorithms, 24, 25, 26
  Cauchy mutation, 32
  component-based exploration and, 48
  continuous, 26–27
  development of, 8, 24
  evolutionary computation theory, 31–32
  for finite-state machines (FSMs), 24, 32
  meta EP, 25, 26
  mutation operator, 26, 32
  for program parse trees, 27
  recommended readings, 65
  Rmeta EP, 25, 26
  search space in, 25
  self-adaptation in, 25
  solution space in, 25
  standard EP, 25
  tournament selection in, 25–26
  TSP, 25
evolutionary search
  algorithms, 5–7
  heuristic search vs., 4
  parallel nature of, 7
  previous solutions used in, 5
  search space, 4–5
evolvable hardware in Sound Gallery, 224–228
evolved aesthetic objects, 136–137
evolved bactericidal peptides, 525–545
  alphahelical peptides, 528
  amphiphilic peptides, 528, 529
  bactericidal activity formula, 531
  computer-aided molecular design (CAMD) approach, 525–526, 527, 528, 542
  conclusions, 542–543
  design cycle, 526–528
  design rules, 529
  evolution, 536–537
  experimental measures and modeling techniques, 529–536
  forward modeling, 526, 527
  GA for, 536–537
  Hamming distance, 540–541
  hypothesis refinement, 427
  laterally amphipathic peptides, 528, 529, 534
  measured log kills, 536, 537
  mechanism of action, 528–529, 535
  model inversion/optimization, 526, 527
  molecular modeling, 531–534
  molecular parameters, 533
  multi-layer perceptron (MLP), 535
  neural networks, 526, 534–536, 538–539
  patent application, 526, 537–543
  peptides defined, 525
  primary peptide sequences, 532
  quantitative structure activity relationships (QSAR) model, 526
  synthesize-and-test method, 525, 526–527
  test bacteria, 529–530
  variables for mechanism of action, 535
evolved images as art, 360–364
  constraints of color and form, 361–363
  found vs. created art, 360–361
  organic images, 364
  visual cortex responses, 363–364
  See also art using GP
explicit embryogeny, 50
explicit encoding, 87–91
exploration
  breadth-first vs. depth-first, 152
  component-based, 45–46, 48, 49
  of search space, creativity in, 60
external embryogeny, 50

F

faces, evolutionary generation of. *See* evolutionary generation of faces
feedback mechanism, as evolution requirement, 148
Ficici, Sevan
  chapter by, 511–523
  See also physical robotics
field-programmable gate arrays (FPGAs), 447
filling shapes in Escher Evolver project, 429, 431, 432
filters
  defined, 283–284
  elliptic lowpass filter application of GP, 287–288
  ladder filter application of GP, 285–286
  M-derived half section circuit application of GP, 286–287
  topology, sizing, placement, and routing using GP, 288–290
fingerprinting, 445, 462–464
finite-state machines (FSMs), EP for, 24, 32
Fish, Stanley, 236
fitness
  of animats, 376–377
  in collaborative evolutionary algorithms, 40

  covariance and selection theorem for, 27–29
  EAs evaluation of, 7–8
  ES evaluation of, 22
  in GenJam, 170–171, 185–186
  GP evaluation, 280–281
  human guidance in GA, 319
  landscapes, 8
  in learning classifier systems (LCSs), 478
  in physical robotics, 516
  relative to population members, 80–81
  in Rooke's GP art, 341
  in Sound Gallery, 224
  in Vox Populi, 211–216, 218
  See also evaluation; human guidance
fitness functions
  for coevolutionary processes, 80
  competition for limited resources and, 80–81
  in creative evolution framework, 51
  defined, 8
  L-system, 303, 304–305, 306–307, 308, 309
  multimodal, 8
  for optimization, 79–80
  in Sound Gallery, 241–242, 243, 245–247
  user input for, 51–52
fitness landscape in design process context, 253–254
fitness value curve, 32–33
fixation on solutions, 322
Flemming, U., 3
flexibility of evolution-based algorithms, 9
Flock, The, 237
Floreano, D., 80
Fogel, David, 8, 10, 25, 31
Fogel, Lawrence, 8, 24, 25, 176
FORM software, 131
formal matrices in Argenìa, 112
Formsynth program, 232
forward modeling for evolved bactericidal peptides, 526, 527
"Fossil Record" book, 10
found vs. created art, 360–361
FPGAs (field-programmable gate arrays), 447
Fracint, 354
fractal functions in Rooke's work, 132, 138
fractals, genetic. *See* genetic fractals
framework for creative evolution, 46–52
  elements of, 46–47
  embryogeny, 49–50
  evolutionary algorithm, 47–48
  fitness functions, 51
  genotype representation, 48–49
  human input, 51–52
  phenotype representation, 50–51
Framsticks simulator, 512
Frazer, John, 37, 251–252, 316
  chapter by, 252–274
  See also generative evolutionary paradigm
freedom of rotation (öi) in Yagi-Uda antenna, 493, 494, 499–500
Freud, Sigmund, 389, 390
Friedberg, R. M., 10
Friedman, George, 10
Frowd, Charlie D., 338
  chapter by, 409–423
  See also evolutionary generation of faces
FSMs (finite-state machines), EP for, 24, 32
functionally complete bases, 445, 459
functions
  automatically defined (ADFs) in GP, 19–20, 49
  iterative, in the complex plane, 347–348
  in Rooke's GP art, 340, 347–348
  See also fitness functions
Fundamental Question, The (TFQ), 444, 457
Funes, Pablo
  chapter by, 511–523
  See also physical robotics
Furuta, H., 40

G

Gabora, Liane, 78, 157
  chapter by, 147–161
  See also beer can theory of creativity
GADES (genetic algorithm designer), 41–42, 56–57, 58
GAEA (generalized architecture for evolutionary algorithms), 33–35
gain in antenna design
  crooked-wire genetic antenna, 502
  defined, 490
  treelike genetic antennas, 507
  Yagi-Uda antenna, 493, 494
gain pattern in antenna design, 490–491, 498
galSim program, 242, 243
"Game of Life," 392, 396
Gardner, Martin, 392
GAs. *See* genetic algorithms (GAs)
Gaudi, Antonio, 259
Gaussian mutation in EP, 32
GenBebop, 233

generalized architecture for evolutionary algorithms (GAEA), 33–35
generalized evolutionary algorithms, 33–35
generating creative results, 42–46
  component-based representations, 44–46
  design fixation and, 43
  exploration, 45–46
  making evolution more creative, 43–44
  optimization, 44–45
  removing constraints, 43–44
generative approach of Argenìa, 109, 110
generative architecture design using L-systems. *See* Lindenmayer systems for architecture design in GP
generative evolutionary paradigm
  chart of, 256
  code scripts in, 255, 257
  coevolution in, 269
  concept seeding approach, 259–260
  conclusions, 271–272
  convergent evolution by natural selection in, 255–256
  convergent/divergent evolution in, 256
  crossover operator, 255, 258
  defined, 255
  design schemas, 259
  epigenetic development in, 255, 257, 267
  evaluation in, 255, 257–258
  evolutionary combinations possible, 256–257
  evolving seeds, 269–271, 272
  genetic code in, 255, 257
  hypothesis underlying, 254
  logic fields, 266–269
  machine-assisted creativity and, 271–272
  motes, 265, 266–267
  mutation, 255, 258
  problems with, 257–258
  representations in, 255, 257
  Reptile system, 260–264
  selection in, 255, 256, 257–258
  Universal State Space Modeler, 264–266
  See also Reptile system; Universal State Space Modeler
genes
  ES strategy parameters and, 21
  in GAs, 12
  linkage equilibrium in, 153
  pioneer, in generative evolutionary paradigm, 257
genetic algorithm designer (GADES), 41–42, 56–57, 58
genetic algorithms (GAs), 10–16
  advanced, 14–16
  basic principles, 14
  canonical or simple, 12–14
  CHC method, 324–328
  circuit design discovery using, 443–466
  common features, 14
  component-based exploration in, 45, 48
  crossover operator, 12, 13
  as design tools, 315–316
  development of, 8, 10–11
  ES vs., 20–21, 23
  for evolving bactericidal peptides, 536–537
  forgiving nature of, 11
  genes, 12
  genetic representation-based problems in, 49
  genotypes, 11, 48
  GP vs., 16–17
  Intelligent Genetic Design Tool, 315–335
  in LCSs, 472–473
  local search vs., 14
  machine learning context, 471–479
  mating pool, 12
  in MAV, 151
  with multistate three-dimensional cellular automata, 268
  mutation operator, 12–13
  natural evolution similarity of, 11
  objective function independence of, 319
  overview, 255
  phenotypes, 11–12, 48
  premature convergence in, 14–15
  randomness in, 13, 318
  recombination in, 318
  recommended readings, 64
  reproduction, 14
  search space, 11
  solution space, 11
  Sound Gallery algorithms, 237–242
  for test pilot creativity simulation, 467–486
  wide search of design space by, 318
  wide use of, 11
  See also antenna design using GA; circuit design discovery using GA; Intelligent Genetic Design Tool; test pilot creativity simulation
Genetic and Evolutionary Computation Conference, 281
genetic antennas. *See* antenna design using GA

genetic code in generative evolutionary paradigm, 255, 257
genetic cross dissolve, 358–359
genetic engineering, GAs with, 16
genetic fractals, 351–358
  in the complex plane, 357, 358
  Fracint-inspired, 354, 355–356
  genomes, 351–352, 354, 356
  second-order subtleties of orbit trajectories, 357, 358
  Skaters, 131, 136, 351
  Through Caverns Measureless, 351, 353
  Winter Solstice, 354, 355
  See also art using GP
Genetic Images, 130–131, 132, 135
genetic programming (GP), 16–20
  advanced, 19–20
  art using, 339–351
  automatically defined functions (ADFs), 19–20, 49
  automatically defined iterations (ADIs), 20, 49
  automatically defined loops (ADLs), 20, 49
  automatically defined recursions (ADRs), 20
  automatically defined stores (ADS), 20
  biological metaphor underlying, 281–282
  biologically inspired computation, 275–298
  bloat in, 19
  circuit synthesis applications, 283–290
  for circuit topology, sizing, placement, and routing, 288–290
  circuit-constructing program tree functions, 284
  component-based exploration in, 45, 48, 49
  conditional statements in, 19
  controller synthesis applications, 290–294
  crossover operator, 17, 18
  development of, 8, 16
  dominance and, 19
  editing operator, 19
  elliptic lowpass filter application, 287–288
  encapsulation operator, 19
  evaluation in, 280
  excessive disruption in, 20
  GAs vs., 16–17
  genetic representation-based problems in, 49
  hierarchical tree structures in, 16–17
  human-competitive results from, 277–280
  junk code generated by, 18–19
  ladder filter application, 285–286
  Lindenmayer systems, 299–313
  M-derived half section circuit application, 286–287
  mutation operator, 17, 281
  overview, 280–282
  permutation operator, 19
  populations, 20, 280–281
  probabilistic steps, 281
  recommended readings, 64, 281
  schema theorem and, 30–31
  selection in, 280–281
  solutions in, 17
  Turing's ideas and, 282
  two-lag plant controller application, 292–294
  See also art using GP; biologically inspired evolutionary computation; circuit synthesis applications of GP; Lindenmayer systems for architecture design in GP
Genetic Programming and Evolvable Machines journal, 281
Genetic Programming Conference, 281
Genetic Programming: On the Programming of Computers by Means of Natural Selection, 281
Genetic Programming II: Automatic Discovery of Reusable Programs, 281
Genetic Programming III: Darwinian Invention and Problem Solving, 281
Genetic Repair operator in GenJam, 178
genetically evolved algorithmic images. *See* art using GP
GenJam, 165–187
  architecture, 166–168
  artistic positioning of, 165–166
  audience-mediated performance by, 185
  chase chorus (trading fours or eights) with, 168, 179–184, 185–186
  chord progression file, 166–167
  chord-scale mappings, 169, 170
  choruses file, 167
  conclusions, 184–186
  convergence issues in, 170–172, 178–179
  crossover operator, 173–175
  evaluation function, 168
  fitness in, 170–171, 185–186
  genetic operators and training, 172–175
  Genetic Repair operator, 178
  GenJam Normal Form (GJNF), 182
  hardware for, 166, 179
  harmonic rules and, 170

GenJam *(continued)*
  head sequence, 166
  hexatonic scale implementation, 169
  hold events, 169, 174–175
  Invert operator, 176, 177, 182
  Invert/Reverse operator, 176, 177, 182
  Lick Thinner operator, 178
  measure mutation, 175–177
  measure population file, 168, 169–170, 172, 173
  mentor for, 168, 170–171
  MIDI events handling, 180, 181
  MIDI parameters file, 167
  musically meaningful mutation in, 175–179
  mutation operators, 165, 175–179, 183
  new-note events, 169, 172, 174, 175
  note-on events, 180
  Orphan Phrase operator, 178
  overview, 165–168
  phrase mutation, 177–179
  phrase population file, 168, 170–173
  real-time interaction with, 168, 179–184, 185–186, 235
  replacement in populations, 178–179
  representations, 168–172
  rest events, 169, 174–175
  Reverse operator, 176, 177, 178, 182, 183
  rhythm sequence, 166
  Rotate Right operator, 176–177, 178, 182, 183
  Sequence Phrase operator, 177, 178, 183
  as soloist, 168, 172–173, 185
  Sort operators, 176, 177
  Super Phrase operator, 178
  tournament selection in, 173
  Transpose operator, 175–176, 182
  "tweak" mode, 173
  Virtual Quintet, 165, 166, 184–185
GenJam Normal Form (GJNF), 182
genomes
  Afman image, 343, 345, 346, 347
  Breed project, 140
  digraph, 348
  genetic cross dissolve, 358
  genetic fractals, 351–352, 354, 356
  in Rooke's work, 131–133, 340, 343, 345, 346, 347, 348, 351–352, 354, 356, 358
  Tuboid project, 142
genotypes
  component-based, 48
  in creative evolution framework, 48–49
  cultural analog of, 149
  defined, 11
  embeddedness of, 96–97
  embryogeny vs., 51
  EP and, 48
  ES and, 48
  in example creative application, 52
  in GAs, 11, 48
  L-system, 302, 304
  phenotypes vs., 51, 91
  in Sound Gallery, 233–234
geological time in Rooke's work, 133
Gero, John, 37, 50, 58
Gino-F, 395
GJNF (GenJam Normal Form), 182
global transition function in ChaOs system, 198
goals
  for circuit design discovery using GA, 444
  for crooked-wire genetic antenna design, 502
  dual-goal evolution in L-systems, 309
  in natural vs. artificial evolution, 224–225, 254
  single-goal evolution in L-systems, 305–309
  for test pilot creativity simulation, 469, 482–483
Godel, K., 389
Goldberg, David, 8, 30, 60
Gordon, William, 318, 323
GP. *See* genetic programming (GP)
grammars
  creativity and, 3
  overview, 3
  in Rooke's work, 131–132
  shape, 50
granular synthesis, 194–195
Gregorian chant, 206
Gregory I, Pope, 206
ground plane of antennas, 490
ground shapes in Escher Evolver project, 427, 430
Gudwin, Ricardo
  chapter by, 205–221
  See also Vox Populi
Guido d'Arezzo, 205, 206

H

Haeckel, Ernst, 361
Hagglund, T., 291
hammerhead turn (PST), 470

Hamming distance for peptides, 540–541
Hancock, Peter J. B., 338
  chapter by, 409–423
  See also evolutionary generation of faces
hardware
  artists, 379
  for evolutionary generation of faces, 415
  for GenJam Virtual Quintet, 166, 179
  LEGO MINDSTORMS robots, 383–384
  for Reptile system, 261
  for Sound Gallery, evolvable, 224–228
harmony
  defined, 214
  fuzzy set for, 212–213
  GenJam and harmonic rules, 170
  Vox Populi fitness criteria, 214–215
Harvey, I., 15
Haydn, Joseph, 206
head sequence for GenJam, 166
Hébert, Jean-Pierre, 362
helicopter gun attack, 470
Helpless Robot, The, 236
Herbst maneuver, 470, 482
Herbst, Wolfgang, 470
heuristic search, 4
hexatonic scales, GenJam implementation, 169
hGAs (hybrid GAs), 15
Hidden Order, 97
Hidden Order of Art, 389–390
hierarchical tree structures in GP, 16–17
higher-level conventions in musical evolution, 194
Higuchi, T., 447
hill-climbing algorithm in Sound Gallery, 238–240
Hillier, B., 306, 311
Hillis, W., 80
hold events in GenJam, 169, 174–175
Holland, John, 8, 10, 29, 61, 80, 97, 100–101, 255, 280
Hopkins, John "Hoppy," 393
Hornby, Greg
  chapter by, 511–523
  See also physical robotics
Houghten, R. A., 539, 541
H-plane of Yagi-Uda antenna, 493, 497, 498
human guidance
  for aesthetic evaluation, 39–41, 42
  in Artificial Painter, 377
  in collaborative evolutionary algorithms, 39–41
  in creative evolution framework, 51–52
  in evolutionary algorithms, 377
  in evolutionary generation of faces, 412, 414
  in GA fitness information, 319
  in GenJam, 168, 170–171
  in Reptile system evaluation, 264
  in Sound Gallery, 224
human-competitive results
  criteria for, 277
  from GP, 277–280
Husbands, P., 39
hybrid GAs (hGAs), 15
hyperparasites, 85

I

I Ching, 391
Iba, H., 447
idea-projects (metaprojects) with Argenìa, 109
ideas, evolution of, 149, 150
identity, evolutionary design structure and, 110, 125
IGAs. *See* Interactive GAs (IGAs)
Ihnatowicz, Edward, 235, 237, 395
IIGAs. *See* Injection Island GAs (IIGAs)
ILLIAC suite, 206
illogical nature of creativity and evolution, 294–296
image hyperspace in Rooke's work, 133
imitation
  in animal behavior, 156
  innovation ratio to, 151
  invention vs., 155–156
  memes acquired by, 151
  in musical evolution, 193
immaterial artists
  electronic, 374–375
  overview, 370–371
implicit embryogeny, 50
implicit encoding, 87–91
improvement, as creative, 59
In the Beginning, 131, 136
  See also Color Plate 4
incidence matrix of IGDT, 324
industrial object design in Argenìa, 121–123, 124
Infinite Permutations V1, 402, 403
Infinite Permutations V2, 402
inheritance
  in EA design space, 33–34
  in Universal Darwinism, 33

initialization in EA design space, 34
initializer routine for treelike genetic antennas, 506
Injection Island GAs (IIGAs)
  described, 16
  two-phase hill-climbing algorithm (Sound Gallery), 238–240
innovation
  antenna design using GA, 487–510
  creativity vs., 60–61
  evolutionary techniques in physical robotics, 511–523
  evolved bactericidal peptides, 525–545
  GA as discovery engine, 443–466
  imitation ratio to, 151
  memes acquired by, 151
  nonrandom, in humans, 152
  test pilot creativity simulation, 467–486
  See also antenna design using GA; circuit design discovery using GA; evolved bactericidal peptides; physical robotics; test pilot creativity simulation
Institute of Contemporary Art, 392
integral of the time-weighted absolute error (ITAE), 292
integration, as fitness function for architecture, 306–307, 308
intelligence
  correlation with creativity, 159
  machine, 282
  See also artificial intelligence (AI)
Intelligent Genetic Design Tool, 315–335
  analysis tools vs. design tools, 316–317
  characteristics, 320–323
  CHC method in, 324–328
  chromosomes, 324
  conclusions, 335
  defined, 320–322
  design criteria altered by, 322
  evolutionary system advantages in design, 317–319
  examples vs. objectives in, 320
  as "exploitative," 322
  historical background, 316
  incidence matrices, 324, 325
  initial generation, 329–330
  initial generation with designer selection/interaction, 330–331
  inner layer, 324
  as "intelligent," 321
  mechanics, 323–328
  need for, 315
  operation, 328–334
  outer layer, 323–324
  preprogramming avoided by, 321
  problem definition, 328–329
  relation to design process, 320–321
  second-generation designer interaction, 332, 333
  second-generation response, 331–332
  third generation, 332–334
  traditional analysis and design programs vs., 320–322
"Intelligent Machines," 282
interactive, adaptive, and autonomous artworks, 235–237
interactive evaluation. *See* human guidance
Interactive GAs (IGAs)
  described, 15
  GenJam algorithms vs., 171
  suitability for musical composition, 207
interactive genetic art, 234–235
interactive pad control in Vox Populi, 217–218
interface of Vox Populi, 216–218
"International Interactive Genetic Art" project, 131
intersection of notes, 213, 214
intrinsic adaptation, 91–92
introns (junk code), 18–19
invention
  creativity and inventiveness, 276–280
  cultural origins and, 155
  imitation vs., 155–156
invention machine, 48
Invert operator in GenJam, 176, 177, 182
Invert/Reverse operator in GenJam, 176, 177, 182
invisible landscape, 133, 137
isospatial model. *See* Universal State Space Modeler
ITAE (integral of the time-weighted absolute error), 292
iterated prisoners' dilemma (IPD), 480–481
iterations
  automatically defined (ADIs), in GP, 20, 49
  iterative functions in Rooke's GP art, 347–348
Iwata, M., 447

J

Jackson, Helen, 252
  chapter by, 299–313

See also Lindenmayer systems for architecture design in GP
Jain, A. K., 450
Jakobi, N., 51
Janssen, P., 253–254
jazz improvisation software. *See* GenJam
Job, Dominic
  chapter by, 443–466
  See also circuit design discovery using GA
Johnston, V. S., 412
Jolly, Julia, 223
Jones 1942 controller patent, 293
Jones, Harry, 293
Journal of Musicological Research, 194
junk code, 18–19

K

Kac, E., 236
Kalganova, Tatiana
  chapter by, 443–466
  See also circuit design discovery using GA
Kasparov, Gary, 4
Kauffman, S. A., 156, 539
Kay, Alan, 388
KBSs (knowledge-based systems), 3
Keane, Martin A.
  chapter by, 275–298
  See also biologically inspired evolutionary computation
Kelly, Kevin, 134, 136, 137–138
Kern, Jerome, 179
Kinetics Group, 395
Kleiweg, Peter, 131
Kluwer Academic Publishers, 281
Knight, T. W., 3
knowledge-based systems (KBSs), 3
Koch curves, L-system examples using, 300, 301
Koning, H., 3
Koza, John, 8, 16, 17, 38–39, 48, 50, 252, 285, 340
  chapter by, 275–298
  See also biologically inspired evolutionary computation

L

Labout, H., 427
ladder filter application of GP, 285–286
Lansdown, John, 395, 396
laterally amphipathic peptides, 528, 529, 534
  See also evolved bactericidal peptides
Latham, William, 37–38, 130, 131, 133, 134–135, 137, 138, 139, 142, 151, 231, 232, 233, 316
Laws of Form, 389, 390–391, 402
LCSs. *See* learning classifier systems (LCSs)
Le Corbusier, 259
learning classifier systems (LCSs), 471–479
  action encoding, 474–476
  baseline test matrix, 478–479
  classifiers, 472
  combat simulation, 474
  combat simulation starting conditions, 478–479
  conflict resolution (CR), 472
  credit allocation (CA), 473, 476, 484–485
  Defensive (DEF) start condition, 478, 479
  design considerations, 483–485
  detectors and classifier conditions, 474, 475
  effectiveness measures, 476–477
  effectors and classifier actions, 474–476
  environmental state encoding, 474, 475
  epochal schemes, 485
  fitness, 478
  GA activation, 472–473, 477–478
  High-Speed Head-On Pass (HSHP) start condition, 478, 479
  High-Speed Line Abreast (HSLA) start condition, 478, 479
  match-and-act, 476
  Monte Carlo RL methods, 484–485
  Offensive (OFF) start condition, 478, 479
  overview, 472–473
  Q-learning, 484–485
  Slow-Speed Line Abreast (SSLA) start condition, 478, 479
  "smoothed" interpretation of actions in, 484
LEGO
  MINDSTORMS robots, 383–384
  simulators for, 516
Leibniz, G. W., 391
level formation, musical evolution and, 194
Levitan, B., 539
Lick Thinner operator in GenJam, 178
life
  Muller's definition of, 91
  origin of, 156
  Ray's definition of, 86
  See also artificial life models

Lin, G., 32
Linden, Derek, 442, 493, 495
  chapter by, 487–510
  See also antenna design using GA
Lindenmayer systems for architecture design in GP, 299–313
  alphabet, 300, 302, 303
  architectural representation, 309–311
  architecture design as multigoal task, 307
  artificial selection, 302–305
  axiom, 300
  coevolution, 310
  conclusions, 311–312
  dual-goal evolution, 309
  evolution toward low i-values, 307, 308
  examples, 300–301
  explicit embryogeny in, 50
  eyeball test as fitness function, 303, 304–305
  fitness functions, 303, 304–305, 306–307, 308, 309
  genotypes, 302, 304
  integration as fitness function, 306–307, 308
  interlinked systems in, 310–311
  isospatial grid for phenotypes, 302, 303
  naïve architectural form representation, 310–311
  overview, 299–302
  phenotypes, 302, 303, 304, 305–306
  rewrite rules, 300
  search space in, 309
  set of productions, 300, 302
  single-goal evolution, 305–309
  spatial embryology in, 302, 303, 304, 311
  structural stability as fitness function, 307, 309
linear polarization in antenna design, 492
linkage equilibrium, 153
Lipnitskaya, Natalia
  chapter by, 443–466
  See also circuit design discovery using GA
Lipson, Hod
  chapter by, 511–523
  See also physical robotics
LISP
  for L-systems, 300
  for Rooke's GP art, 340
Liu, Y., 31, 32
local search, 6, 14
logic fields, 266–269
logic structures in Argenìa, 111, 118
Lombardo, Sergio, 371
loops, automatically defined (ADLs), in GP, 20, 49
loudness, 208
lowpass filters
  elliptic filter application of GP, 287–288
  ladder filter application of GP, 285–286
  M-derived half section circuit application of GP, 286–287
  overview, 283–284
  topology, sizing, placement, and routing using GP, 288–290
L-systems. *See* Lindenmayer systems for architecture design in GP
Luigi Russolo competition (Italy), 202
Lund, Henrik Hautop, 338
  chapter by, 367–385
  See also electronic art

M

machine intelligence, 282
machine learning, genetics based, 471–479
machine-assisted creativity, 271–272
MacKenzie, Bob, 157
MacKenzie, Doug, 157
Mackintosh, Charles Rennie, 259
Macready, W. G., 539
Maeda, K., 40
Magic Flute, The, 210
Maloy, W. L., 539, 541
"Mandala," 393
Mandelbrot, Benoit, 394, 396
Mandelbrot sets
  in Fracint, 354
  in genetic fractal genomes, 354
  iterative functions for, 347–348
  orbit trajectories, 358
Manzolli, Jônatas, 210
  chapter by, 205–221
  See also Vox Populi
mapping
  in ChaOs system, 199–201
  defined, 34
  embryogeny as, 49–50
  in GAEA, 34
Margulis, Lyn, 143
Markov chain analysis
  for convergence theory in EAs, 31
  deceptive and hard problems identified by, 30
Martini, Simone, 372
material artists
  electronic, 373–374
  overview, 371–372

materiality
  fundamental resources, 98–99
  matter, 99
  open-ended evolution and, 98–100
  predator-prey relationships and, 98
  Tierra-like platforms and, 99
"Mathematical Recreations," 392
Mathews, Max, 206
mating pool, in GAs, 12
MAV (Meme and Variations) computer model, 151–152
MAX circuit symbol, 450
MAX gates, 450
McKenna, Terence, 133, 137
McMullin, Barry, 93
M-derived half section circuit application of GP, 286–287
mean-squared error (MSE), as face generation performance measure, 418–421
measure mutation in GenJam, 175–177
measure population file for GenJam, 168, 169–170, 172, 173
medieval town (a la Giotto) project in Argenìa, 114–116
Mehra, R. K.
  chapter by, 467–486
  See also test pilot creativity simulation
melodic control in Vox Populi, 216
melodic fitness in Vox Populi, 214
melody, defined for Vox Populi, 208
Meme and Variations (MAV) computer model, 151–152
memes
  acquisition of new memes, 151
  reductionistic overtones of, 149
memory, worldview and, 157
memory stores, automatically defined (ADS) in GP, 20
mental set, 153
mentor for GenJam, 168, 170–171
messy GAs (mGAs), 15
meta EP, 25, 26
metacreation, 134–135
metaprojects (idea-projects) with Argenìa, 109
mGAs (messy GAs), 15
Michielssen, E., 488
MIDI
  GenJam event handling, 180, 181
  GenJam parameters file, 167
  in Vox Populi, 208, 209
Miller, G. F., 80
Miller, Julian, 38, 441
  chapter by, 443–466
  See also circuit design discovery using GA
Milner, Marion, 389
MIN circuit symbol, 450
MIN gates, 450
Miranda, Eduardo Reck, 164, 199
  chapter by, 189–204
  See also ChaOs system; music in virtual worlds
MLP (multi-layer perceptron) neural networks, 535
model inversion/optimization for evolved bactericidal peptides, 526, 527
MODPRODUCT circuit symbol, 450
MODPRODUCT gates, 450
MODSUM circuit symbol, 450
MODSUM gates, 450
modular circuit construction, 444
MOGAs (multiobjective GAs), 15
molecular modeling for evolved bactericidal peptides, 531–534
momentum of evolutionary creative process, 132, 135
Mondada, F., 80
Mondrian Evolver, 425–426
Mondrian, Piet, 260, 425
Monte Carlo optimization, 536, 537
Monte Carlo RL methods, 484–485
Moore, R. F., 206, 207
Moroni, Artemis, 164
  chapter by, 205–221
  See also Vox Populi
morphogenesis
  in *Breed* project, 140
  in *Tuboid* project, 142
motes, in Universal State Space Modeler, 265, 266–267
Mount, John, 131, 234
moving, in GAEA, 34
Mozart, Wolfgang, 206, 210
MSE (mean-squared error), as face generation performance measure, 418–421
Muller, H. J., 91
multi-layer perceptron (MLP) neural networks, 535
multimedia urban stand (Milan) project in Argenìa, 119, 121
multimodal fitness function, 8
multiobjective GAs (MOGAs), 15
multipath problems in antenna design, 501
multiple-valued circuit symbols, 449–450
multiple-valued one-digit adder with carry circuit, 459–461

Musgrave, Kenneth, 131
music
  EAs and, 232–234
  Schoenberg's definition of, 223
  See also GenJam; music in virtual worlds; Sound Gallery; Vox Populi
music in virtual worlds, 189–204
  ChaOs system, 196–202
  coevolution, 192
  commentary on results, 201–202
  conclusions, 202
  evolutionary modeling, 190–194
  evolving sound with cellular automata, 194–201
  fundamental mechanisms for studying origins and evolution of, 190, 191–194
  level formation, 194
  self-organization, 192–194
  theoretical background, 190–194
  transformation and selection, 191–192
  See also ChaOs system
"Music of Changes," 232
musical fitness in Vox Populi, 215–216
mutation-driven GAs, 15
mutations
  advantages for design, 318
  breeding aesthetic objects and, 135–136
  in EP, 26, 32
  in ES, 21, 22, 23
  GA operator, 12–13
  Gaussian, in Escher Evolver project, 433
  in generative evolutionary paradigm, 255, 258
  GenJam operators, 165, 175–179, 183
  GP operator, 17, 281
  proto-DNA and, 104
  in Reptile system, 262, 264
  in Rooke's GP art, 340–341
  schema theorem and, 30
  as variation source, 149
  in von Neuman's architecture, 83
  in Vox Populi, 209
Mutator program, 232
MUX circuit symbols, 449
MUX gates, 446, 449, 451–452
My Gasket, 404

N

Nabuurs, R.
  chapter by, 425–439
  See also Escher Evolver project
natural evolution
  artificial evolution vs., 224–225
  computer evolution vs., 27
  as creative, 56, 58–62
  creativity of, 2
  dominance in, 19
  evolutionary algorithms and, 27
  GAs similarity to, 11
  as mentor for evolutionary computation, 2
  objectives lacking in, 224, 254
  Universal Darwinism theory, 33
Natural History Museum (London), 130
NEC2 (Numerical Electromagnetics Code, Version 2), 490, 492–493, 506, 507
negative feedback amplifier, 295
Nelson, Gary Lee, 233, 234
netlist for circuits, 283
neural networks
  for evolving bactericidal peptides, 526, 534–536, 538–539
  multi-layer perceptron (MLP) networks, 535
  physical robotics controllers, 514, 518
Neural Works Professional II, 535
new-note events in GenJam, 169, 172, 174, 175
niching, GAs with, 15
nightclub, as Sound Gallery setting, 230
NN. *See* neural networks
Nolfi, S., 80
normal distribution in ES mutation, 23
NOT circuit symbol, 450
NOT gates, 446, 447, 450
"Not Only Computing—Also Art," 396
note-on events in GenJam, 180
Notting Hill, 131
Nova Express, 392
Novak, Marcos, 364
N-point crossover in Escher Evolver project, 432–433
number sign (#) as don't care symbol, 472
Numerical Electromagnetics Code, Version 2 (NEC2), 490, 492–493, 506, 507
Nureyev, Rudolf, 148, 154

O

objective function
  GA independence from, 319
  for Yagi-Uda antenna, 496
objectives. *See* goals
octave control in Vox Populi, 217

"Olivine Trees," 202
On the Origin of Species, 98, 256
one-bit adder circuit, 450–452
  multiple-valued, with carry, 459–461
open-ended evolution
  ability to perform other tasks and, 91–93
  conclusions, 104
  creativity and, 79–82
  design issues, 82–100
  embeddedness of individuals and richness of interactions and, 93–98
  environmental properties and, 100–101
  implicit vs. explicit encoding, 87–91
  intrinsic adaptation needed for, 91–92
  materiality and, 98–100
  specification for, 100–104
  Tierra platform and, 84–87
  von Neumann's architecture for self-reproduction and, 82–84
  Waddington's paradigm for evolutionary process, 101–104
operator adaptation in GAEA, 35
operator rates in GAEA, 35
operons, GP conditional statements and, 19
optimization
  antenna design problems, 487–488
  component-based representation and, 44, 45–46
  defined, 4
  fitness functions for, 79–80
  fixed-length parameterization and, 44–45, 46
  GA design tools for, 315
  IGDT vs., 320
  Traveling Salesman Problem (TSP), 241
  Yagi-Uda antenna, 496, 499–500
OR circuit symbols, 449
OR gates, 446, 447, 450, 451
orchestra control in Vox Populi, 217
order of schema, 29
Orgel, L. E., 101
Orphan Phrase operator in GenJam, 178
O'Toole, A. J., 412
Ottawa Citizen, 147
out-of-plane maneuvering, 470
Owens, A. J., 24, 176

P

Packard, N. H., 80
Pagliarini, Luigi, 338
  chapter by, 367–385
  See also electronic art
Palladio, Andrea, 260
paper fall application, 52–54
paradigm shift, computational arts and, 388
parallel GAs, 15
parameter control in Vox Populi, 216–218
parameter space in Rooke's work, 133
parameterization
  component-based representation vs., 44, 45–46
  fixed-length, optimization and, 44–45, 46
  removing constraints and, 43–44
parasites with Tierra platform, 85
parasitics in antenna design, 490
Parker, Charlie, 179
Parmee, Ian, 37, 43
Pask, Gordon, 235
passband of filters, 284
Patel, Shail, 442, 538, 541
  chapter by, 525–545
  See also evolved bactericidal peptides
patents
  Callender and Stevenson 1939 PID controller, 291
  Campbell 1917 ladder filter, 285–286
  Cauer 1934–1936 elliptic filter, 287–288
  criteria for (U.S.), 286, 294
  for evolved bactericidal peptides, 526, 537–543
  for genetic antennas, 504, 505
  illogical step required for, 294
  Jones 1942 controller, 293
  patent spaces, 539–542
  Zobel 1925 *M*-derived half section, 286–287
patinas in Argenìa, 112
Pattee, Howard, 80, 86–87, 96, 103
pattern of information, as requirement for evolution, 148
PCA (principle components analysis), 409
Peirce, Charles Saunders, 389
Penny, Simon, 236–237
people
  design problems regarding, 36–37
  See also human guidance
peptides
  alphahelical, 528
  amphiphilic, 528, 529
  defined, 525
  Hamming distance for, 540–541
  laterally amphipathic, 528, 529, 534
  See also evolved bactericidal peptides
permutation operator in GP, 19

Peter and the Wolf, 210
Petit Mal, 237
phenotypes
  in creative evolution framework, 50–51
  cultural analog of, 149
  defined, 11
  DNA and, 271
  embryogeny vs., 51
  EP and, 48
  ES and, 48
  evolution of new modalities by, 103
  in example creative application, 52
  in GAs, 11–12, 48
  genotypes vs., 51, 91
  interaction with environment and, 88
  isospatial grid for, 302, 303
  L-system, 302, 303, 304
  proto-DNA properties, 90–91
photo-fit composite systems, 409
photography, algorithmic image breeding compared to, 361
phrase mutation in GenJam, 177–179
phrase population file for GenJam, 168, 170–173
physical robotics, 511–523
  autonomous robotics issues, 511–512
  buildable simulation, 515–516
  coevolution, 512–513, 520
  conclusions, 519–520
  continuous body deformation, methods for, 514–515
  controllers, 512, 513, 514, 518–519
  embodied evolution, 518–519
  evolution and construction of electromechanical systems, 517
  evolution in simulation, 514–515
  fitness, 516
  integrating body and brain design, 512
  long-term power issues, 518
  neural network controllers, 514, 518
  real-world adaptation, fine-tuning, 518–519
  reprogramming issues, 518
  research thrusts, 513–514
  robot learning experiments, 518–519
  technologies aiding, 512
Piazza Navona, 113
Picasso repainting with Argenìa, 123, 124
Pickover, Clifford, 347
PID controller synthesis applications of GP
  overview, 290–292
  for two-lag plant, 292–294
pioneer genes in generative evolutionary paradigm, 257
pitch, 208
placement of circuits, automatic synthesis using GP, 289–290
plane transformations in Escher Evolver project, 427–428, 429, 432
polarization in antenna design
  crooked-wire genetic antenna, 502
  defined, 492
  Yagi-Uda antenna, 494
Pollack, Jordan, 39, 442
  chapter by, 511–523
  See also physical robotics
populations
  covariance and selection theorem for, 27–29
  in EP, 26
  in ES, 22
  fitness relative to members of, 80–81
  in GAs, 11
  in GPs, 20, 280–281
  initialization in EA, 34
  moving in, 34
  of solutions, advantages for design, 317–318, 322
post-stall technology (PST), 470–471
Poundstone, W., 396
Prado Museum (Madrid) project in Argenìa, 119, 120
predator-prey relationships, competition and, 98
premature convergence
  avoiding in GAs, 15–16
  defined, 14–15
Price, Cedric, 260
Price, G. R., 27
Prince, George, 318
principle components analysis (PCA), 409
principle extraction, 443, 444–445, 462–464
printmaking, algorithmic images and, 350–351
probabilistic steps in GP, 281
program parse trees, EP for, 27
Prokofiev, Sergey, 210
proto-DNA
  defined, 89
  environment for, 104
  explicit vs. implicit encoding and, 89–91, 92–93
  mutations and, 104
  phenotypic properties of, 90–91
PST (post-stall technology), 470–471
Psycho-Analysis of Artistic Vision and Hearing, The, 389

Putnam, Jeffrey, 234
Pythagoras, 370, 374

Q

Q-learning, 484–485
QSARs (quantitative structure activity relationships), 526
quadrafilar helix antenna, 501
quality
  beauty as measure of, 125
  See also evaluation
quantitative structure activity relationships (QSARs), 526
quiescent cell state in ChaOs system, 197, 198

R

radix, 449
Rahmat-Samii, Y., 488
random fluctuations, as evolution requirement, 148, 149
randomness in GAs, 13, 318
Ravichandran, B.
  chapter by, 467–486
  See also test pilot creativity simulation
Ray, Tom, 80, 81, 84, 85–86, 93
  See also Tierra (and Tierra-like) platforms
real-time interaction with GenJam, 179–184
Rechenberg, Ingo, 8, 31
recognizability in Argenìa, 109–110, 125
recombination, advantages for design, 318
recommended readings
  AI and creativity, 63–64
  evolution strategies, 64–65
  evolutionary programming, 65
  genetic algorithms, 64
  genetic programming, 64
reconfigurable chips in Sound Gallery, 226–228
recursions, automatically defined (ADRs) in GP, 20
Red Queen effect, 480–481
Reed-Muller (RM) algebra, 445–446
reflector in Yagi-Uda antenna, 493
regulons, GP conditional statements and, 19
Reichardt, Jasia, 392
Religious view of creativity, 62
representations
  architectural, in L-systems, 309–311
  component-based, 44–46, 48–49, 52–54
  creativity as going beyond, 61–62
  in Escher Evolver project, 429
  in generative evolutionary paradigm, 255, 257
  genetic representation-based problems, 49
  in GenJam, 168–172
  genotype, 48–49
  phenotype, 50–51
  as primary for creative evolutionary systems, 48
  for simulators in physical robotics, 515
  space of, for circuit design discovery, 445–447
reproduction
  in EA design space, 33
  in GAs, 14
  in Universal Darwinism, 33
  Vox Populi operator, 209
Reptile system, 260–264
  cultivating seeds, 262, 264
  data structure, 261
  epigenetic process, 262
  evaluation, 264
  mutations, 262, 264
  seeds, 261, 262, 264
  structural units, 260–261
rest events in GenJam, 169, 174–175
restart in GAEA, 35
Reverse operator in GenJam, 176, 177, 178, 182, 183
rewrite rules for L-systems, 300
rhythm sequence for GenJam, 166
rhythmic control in Vox Populi, 217
richness of interactions, open-ended evolution and, 94–98
Rinaldo, Ken, 235, 236, 237
RM (Reed-Muller) algebra, 445–446
Rmeta EP, 25, 26
Roads, Curtis, 194–195
robots
  LEGO MINDSTORMS, 383–384
  physical robotics, 511–523
  Shakey, 512
  See also physical robotics
robustness
  of evolution-based algorithms, 9
  of GAs, 11
Rocha, Luis, 100
Rollins, Sonny, 171
Rooke, Steven, 131, 134, 135, 136–137, 138–139, 143, 337

Rooke, Steven *(continued)*
  case study of, 131–133
  chapter by, 339–365
  In the Beginning by, 131, 136
  Skaters by, 131, 136
  See also art using GP
Rosenman, Michael, 37, 310
Rotate Right operator in GenJam, 176–177, 178, 182, 183
roulette wheel selection in Escher Evolver project, 433
routing circuits, automatic synthesis using GP, 289–290
Rowbottom, Andrew, 131
Rudolph, G., 31

S

SAGA, 15
"SAM—Sound Activated Mobile," 395
Samuel, Arthur, 275
SAND LINES, 402, 403
satisficer, 44
scalability, implicit embryogeny and, 50
Schattschneider, D., 428
schema, 29
schema theorem, 29–31
  future generations and, 30
  GP and, 30–31
Schoenberg, Arnold, 223, 245
Scientific American, 392
Scientist view of creativity, 62
search
  in design process context, 253–254
  evolutionary vs. heuristic, 4
  for genetic library building, 347
  heuristic, 4
  machine intelligence and, 282
  overview, 3–4
  See also evolutionary search
search space
  alternative spaces, 60
  creativity in exploration of, 60
  crooked-wire genetic antenna, 501, 502
  defined, 4
  design problems for creative evolutionary computation, 48–49
  in design process context, 253–254
  in EP, 25
  evolution considered in, 7
  in GAs, 11
  genotype representation as, 48
  in L-systems, 309
  proximity of qualities in, 5
  of representations for circuit design discovery, 445–447
  Universal State Space Modeler, 264–266
  wide search of design space, 318
seeds
  concept seeding approach, 259–260
  evolving, 269–271, 272
  in Reptile system, 261, 262, 264
Segal, Walter, 260
selection
  artificial, 256
  in EA design space, 34
  environmental matching by, 149
  in Escher Evolver project, 433
  in generative evolutionary paradigm, 255, 256, 257–258
  in GP, 280–281
  L-system, artificial, 302–305
  in musical evolution, 191–192
  in Universal Darwinism, 33
selective feedback computer, 10
self-adaptation
  in EP, 25
  in ES, 23
self-organization
  cultural, 193–194
  musical evolution and, 192–194
self-reproduction. *See* architecture for self-reproduction (von Neumann)
semantic closure, 96
seminal computer evolution, 9–10
semiology, 389
sensing systems of Sound Gallery, 230–231
"Senster, The," 235–236, 237, 395
Sequence Phrase operator in GenJam, 177, 178, 183
set of productions for L-systems, 300, 302
sGAs (structured GAs), 15
Shakey the robot, 512
shape grammars, explicit embryogeny in, 50
Sharman, Ken, 237
Sierpiński gasket, 300, 301
simple GAs. *See* canonical (simple) GAs
Sims, Karl, 38, 39, 80, 136, 137, 138, 142, 151, 231, 232, 233, 316, 339, 340, 348, 358, 512, 513
  Genetic Images by, 130–131, 132, 135
simulation. *See* physical robotics; test pilot creativity simulation

single-goal evolution in L-systems, 305–309
sizing circuits
  automatic synthesis using GP, 288–290
  defined, 283
Skaters, 131, 136, 351
  See also Color Plate 5
Slade School of Art, 392
slide immersion, 137
Smith, Maynard, 101, 103
Smith, Robert, 442
  chapter by, 467–486
  See also test pilot creativity simulation
social context of electronics, 368–370
  feedback, 370
  locations of interaction, 368–369
  technological influences on art, 369–370
social evolution, research paradigms and, 189–190
Soddu, Celestino, 37, 78, 310
  chapter by, 109–127
  See also Argenìa
solutions
  as emergent property of EAs, 6–7
  EP solution space, 25
  fixation on, 322
  GA solution space, 11
  in GP, 17
  populations of, 317–318, 322
  previous, in evolutionary search, 5
Sonomorphs, 233, 234
Sort operators in GenJam, 176, 177
soul expression, creativity as, 62
sound attributes, 208
Sound Gallery, 223–250
  aim of, 223
  algorithms, 237–242
  art gallery as setting for, 230
  beach as setting for, 229
  cellular automaton, 242, 243
  conclusions, 247
  contextualization of, 231–237
  encoding scheme, 240
  evolvable hardware, 224–228
  experiment, 242–244
  fitness function, 241–242, 243, 245–247
  gallery setup, 228–230
  galSim program, 242, 243
  genotypes, 233–234
  music, definition for, 223
  nightclub as setting for, 230
  overview, 223–224
  reconfigurable chips, 226–228
  results, 244–247
  sensing systems, 230–231
Sound Hunters, The, 210
sound synthesis, granular, 194–195
source identification circuits, GP synthesis of, 289
space
  design space for treelike genetic antennas, 507
  electronic, 373, 374
  patent spaces, 539–542
  of representations for circuit design discovery, 445–447
  See also search space
spatial embryology in L-systems, 302, 303, 304, 311
speciation, GAs with, 15
species
  in Argenìa, 110, 111, 113, 114
  as identification categories, 110
Spector, L., 233
Spencer-Brown, George, 389, 390–391, 402
Stadler, P. F., 539
standard EP, 25
state space in Universal State Space Modeler, 264–265
steady-state GAs, 15
Steels, Luc, 190, 191
Stevenson, Allan, 291
Stiny, G., 3
stochastic formulae for granular synthesis, 195
stopband of filters, 284
stores, automatically defined (ADS) in GP, 20
Storr, Anthony, 389
Stott, Ian, 538, 541
  chapter by, 525–545
  See also evolved bactericidal peptides
strategy parameters in ES, 21, 22, 23
Strauss, J., 383
structural stability, as fitness function for architecture, 307, 309
structured GAs (sGAs), 15
Strunk, O., 205
subjective creativity, 62
subsumption architecture, 512
Sullivan, Julian, 395
Super Phrase operator in GenJam, 178
Sutton, R. S., 486

symbolic information, evolutionary rise of, 90
System Art Group, 392

T

Tao Te Ching, 391–392
Taura, T., 51
Taylor, Tim, 77
  chapter by, 79–108
  See also open-ended evolution
technology
  computer science and technological culture, 129–130
  feedback from art and the world, 370
  influences on art, 369–370
  physical robotics and, 512
temperature-sensing circuits, GP synthesis of, 288–289
termination in EA design space, 34
test pilot creativity simulation, 467–486
  advantages of, 482–483
  aircraft aerodynamic characteristics and, 482
  Air-to-Air System Performance Evaluation Model (AASPEM), 474, 481
  alternate freeze learning (ALT), 481
  alternate freeze learning with memory (MEM), 481
  approaches, 468–469
  aspect angle for aircraft, 469–470
  cobra maneuver, 470
  conclusions, 483–485
  fighter aircraft maneuvering, 469–471
  genetics-based machine learning, 471–479
  goals, 469, 482–483
  helicopter gun attack, 470
  Herbst maneuver, 470, 482
  iterated prisoners' dilemma (IPD), 480–481
  learning classifier systems (LCSs), 471–479, 483–485
  need for, 467–468
  one-sided learning results, 479–480
  out-of-plane maneuvering, 470
  parallel learning (PAR), 481
  post-stall technology (PST), 470
  PST hammerhead turn, 470
  PST maneuvers, 470–471
  Red Queen effect, 480–481
  two-sided learning results, 480–482
  See also learning classifier systems (LCSs)
TFQ (The Fundamental Question), 444, 457
T-GATE circuit symbol, 450
T-GATEs, 450, 461
theological implications of metacreation, 134–135
Thompson, Adrian, 38, 164, 447
  chapter by, 223–250
  See also Sound Gallery
Thompson, J. L., 412
three-bit multiplier circuit, 457–459
Through Caverns Measureless, 351, 353
Tierra (and Tierra-like) platforms, 84–87
  ad-hoc feel of, 86–87
  analysis in terms of von Neumann's architecture, 87–89
  ancestor program, 85
  competition lacking in, 99
  embeddedness in, 94, 95
  innovations in experiments with, 81
  materiality and, 99
  overview, 84–85
  parasites and hyperparasites with, 85
  philosophy behind, 85–86
  richness of interactions in, 96
tile drawings by Brown, 393–395
timbre, 208
time-optimal controller circuits, GP synthesis of, 289
Titian, 259
Todd, Stephen, 38, 130, 133, 134, 139, 151, 232
topology of a circuit
  automatic synthesis using GP, 288–290
  defined, 281
Totally Reconfigurable Analog Circuit (TRAC), 227–228
tournament selection
  in EP, 25–26
  in GenJam, 173
  in Mondrian Evolver, 433
TPRODUCT circuit symbol, 450
TPRODUCT gates, 450
TRAC020 chip, 227–228
trading fours or eights with GenJam, 168, 179–184, 185–186
Trajan Forum, 113
transfer functions for controllers, 292–293
transformations in musical evolution, 191
Transpose operator in GenJam, 175–176, 182
Traveling Salesman Problem (TSP), 241
treelike genetic antennas, 506–509
  chromosome, 506

  conclusions, 509
  design space, 507
  gains, 507
  initializer routine, 506
  NEC2 and, 506, 507
  resulting antennas, 506–509
Trim, Michael, 393
Truax, Barry, 195
TSP (Traveling Salesman Problem), 241
TSUM circuit symbol, 450
TSUM gates, 450
Tuboid sculptures project, 142–144
Turing, A., 82, 282
two-bit adder circuit, 452–454
two-bit multiplier circuit, 454–457
two-lag plant controller application of GP, 292–294
two-phase hill-climbing algorithm in Sound Gallery, 238–240

U

Uda, S., 493
Ulam, Stanislaw, 84
universal constructor, 84
Universal Darwinism theory, 33
Universal State Space Modeler, 264–266
  coordinate systems, 265–266
  as isospatial modeler, 264
  motes, 265, 266–267
  state space, 264–265
user guidance. *See* human guidance
user interface in Escher Evolver project, 435, 436–437

V

van Hemert, J. I., 426
Van Valen, L., 81
variation
  in culture and biology, 148–149, 150
  in EA design space, 34
  as evolution requirement, 148
  as self-organization requirement, 192
  in Universal Darwinism, 33
Ventrella, Jeffrey, 235
Verstappen, Maria, 139–144
VideoWorks, 402
Vidyamurthy, G., 212
Villa-Lobos, Heitor, 210
Viner, Darrel, 395
virtual worlds, music in. *See* music in virtual worlds
visual arts, EAs and, 231–232
visual cortex, response to evolved images, 363–364
voice range criterion in Vox Populi, 215
voices population in Vox Populi, 209
voltage reference circuits, GP synthesis of, 288–289
Voltage Standing Wave Ratio (VSWR)
  defined, 491–492
  in Yagi-Uda antenna, 494, 497–498
von Buelow, Peter, 252
  chapter by, 315–336
  See also Intelligent Genetic Design Tool
von Neumann, John, 82, 84, 100
  See also architecture for self-reproduction (von Neumann)
Von Zuben, Fernando
  chapter by, 205–221
  See also Vox Populi
Vox Populi, 205–221
  biological control, 216
  conclusions, 219–220
  consonance criterion, 212–214
  crossover operator, 209
  demo of, 219
  evolutionary musical cycle, 208–211
  experiments, 218–219
  fitness displays, 218
  fitness evaluation, 211–216
  genetic cycle, 209–210
  harmonic fitness, 214–215
  historical background, 206–207
  interactive pad control, 217–218
  interface and parameter control, 216–218
  melodic control, 216
  melodic fitness, 214
  musical fitness, 215–216
  mutation operator, 209
  octave control, 217
  orchestra control, 217
  overview, 207
  reproduction operator, 209
  rhythm of the evolution, 210–211
  rhythmic control, 217
  sound attributes, 208
  voice range criterion, 215
  voices population, 209
VSWR (Voltage Standing Wave Ratio)
  defined, 491–492
  in Yagi-Uda antenna, 494, 497–498

W

Waddington, C. H., 91, 101–103, 104
Walsh function analysis, 30
Walsh, M. J., 24, 176
Washabaugh, W., 194
Watanabe, W., 40
Watson, Richard A.
  chapter by, 511–523
  See also physical robotics
"Well-Tempered Clavier, The," 372
White, Norman, 236
Whitelaw, Mitchell, 38, 78
  chapter by, 129–145
  See also breeding aesthetic objects
Whitley, D., 238
Winter Solstice, 354, 355
wire size in antenna design, 492, 507, 509
Wittgenstein, L., 191
Woolf, Ian, 223
Woolf, Sam, 164, 231
  chapter by, 223–250
  See also Sound Gallery
worldview, memory and, 157
Wright, Frank Lloyd, 259
"Wurfelspiel" (Dice Game), 206

X

Xenakis, Iannis, 194
XOR circuit symbols, 449
XOR gates, 449, 450, 452, 457

Y

Yagi, Prof., 493
Yagi-Uda antenna, 493–500
  for Arecibo spherical reflector, 493–498
  beamwidth, 493, 497
  conclusions, 509
  conventional design, 493, 494
  crossover, 494
  directors, 493
  E-plane, 493, 497, 498
  freedom of rotation ($\ddot{o}_i$) in, 493, 494, 499–500
  GA for, 494–497
  gain, 493, 494
  gain patterns, 498
  H-plane, 493, 497, 498
  objective function, 496
  optimization, 496, 499–500
  performance, 497–498
  polarization, 494
  reflector, 493
  VSWR, 494, 497–498
Yao, X., 31, 32
Yu, T., 20

Z

Zasloff, M. A., 537–538, 539, 541
Zobel 1925 *M*-derived half section patent, 286–287
Zobel, Otto, 286